2021-06

ADAPTIVE MACHINE LEARNING-BASED STEGANOGRAPHIC MODEL FOR SUBVERTING CENSORSHIP

Hain, Fritz W.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ADAPTIVE MACHINE LEARNING-BASED STEGANOGRAPHIC MODEL FOR SUBVERTING CENSORSHIP

by

Fritz W. Hain

June 2021

Co-Advisors: Vinnie Monaco
Britta Hale

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
ADAPTIVE MACHINE LEARNING-BASED STEGANOGRAPHIC MODEL FOR SUBVERTING CENSORSHIP

5. FUNDING NUMBERS

6. AUTHOR(S) Fritz W. Hain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Chinese digital censorship is both wide-reaching and harmful to U.S. interests. It prevents the free flow of information and stifles criticism of governance, social movements, and the spread of democratic values. Cryptography has long been a proven solution to secure data in transit, but it has one significant flaw: it is not covert, and thus is easily detectable. We propose an adaptive steganographic model that is both covert and secure and can be implemented with existing messaging platforms. We utilize machine learning to create an adaptive model that modifies a steganographic algorithm, making steganalysis more difficult. We demonstrate the feasibility of ratcheting and otherwise modifying our steganographic model to generate a new solution by reinitializing the final layers of the encoder and decoder models, creating at least 100 unique models with low cross-decoding compatibility (high decoding bit-error). We show the potential for a tiling method of steganographic encoding that exponentially increases encoding capacity but likely carries a higher risk of detection via steganalysis. We show that in the stego-images examined, the induced changes to the images are concentrated at the edges. We demonstrate a significant vulnerability to a novel form of statistical steganalysis in (ML) steganography based on the distribution of bit-errors. Finally, we discuss the necessary steps and key challenges to implementing our model with an existing messaging platform.

14. SUBJECT TERMS steganography, machine learning, privacy, surveillance, messaging, communication, China	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ADAPTIVE MACHINE LEARNING-BASED STEGANOGRAPHIC MODEL FOR SUBVERTING CENSORSHIP

Fritz W. Hain
Lieutenant, United States Navy
BA, Emory University, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by: Vinnie Monaco
Co-Advisor

Britta Hale
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Chinese digital censorship is both wide-reaching and harmful to U.S. interests. It prevents the free flow of information and stifles criticism of governance, social movements, and the spread of democratic values. Cryptography has long been a proven solution to secure data in transit, but it has one significant flaw: it is not covert, and thus is easily detectable. We propose an adaptive steganographic model that is both covert and secure and can be implemented with existing messaging platforms. We utilize machine learning to create an adaptive model that modifies a steganographic algorithm, making steganalysis more difficult. We demonstrate the feasibility of ratcheting and otherwise modifying our steganographic model to generate a new solution by reinitializing the final layers of the encoder and decoder models, creating at least 100 unique models with low cross-decoding compatibility (high decoding bit-error). We show the potential for a tiling method of steganographic encoding that exponentially increases encoding capacity but likely carries a higher risk of detection via steganalysis. We show that in the stego-images examined, the induced changes to the images are concentrated at the edges. We demonstrate a significant vulnerability to a novel form of statistical steganalysis in (ML) steganography based on the distribution of bit-errors. Finally, we discuss the necessary steps and key challenges to implementing our model with an existing messaging platform.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

DoD Department of Defense

MASINT Measurement and Signature Intelligence

NPS Naval Postgraduate School

SRWBR Short Range Wide Band Radio

TCP Transmission Control Protocol

USN U.S. Navy

UDP User Datagram Protocol

USG United States Government

LSB Least Significant Bit

ML Machine Learning

AI Artificial Intelligence

AES Advanced Encryption Standard

ATS Artificial Training Sets

PSNR Pseudorandom Noise Ratio

SSIM Structural Similarity Index Measure

RSA Rivest–Shamir–Adleman

ECC Elliptic-Curve Cryptography

OCR Optical Character Recognition

BPP Bits Per Pixel

CCP Chinese Communist Party

HPC High Power Computing

SGM Star-Generated Model

PRNG Pseudorandom Number Generator

PUF Physical Unclonable Functions

KDF Key Distribution Function

API Application Programming Interface

RS Regular-Singular

WS Weighted-Steganalysis

DIH Difference Image Histogram

SPA Sample Pairs Analysis

DCT Discrete Cosine Transform

FFT Fast Fourier Transform

DFT Discrete Transform

NIST National Institute of Standards and Technology

HUGO Highly Undetectable Steganography

WOW Wavelet Obtained Weights

UNIWARD Universal Wavelet Relative Distortion

SMS Short Message Service

UTF-8 Unicode Transformation Format - 8 Bit

CUDA Compute Unified Device Architecture

P2P Peer to Peer

EOF End Of File

Glossary

Collision Observed decoding bit-error less than .4 for stego-image produced by an encoder from a separate model. In other words, an inter-model bit-error less than .4.

Cost Function A function that calculates the difference between predicted and observed values. In Machine Learning (ML), the cost function often serves to minimize some metric, such as loss, error, or cost.

Intra-model Bit-Error The observed decoding bit-error for one steganographic model.

Inter-model Bit-Error The observed bit-error when one steganographic model decodes a stego-image produced by another steganographic model.

Ratchet-generated Model One of a number of steganographic models created by reinitializing the final layers of the ML model encoder and decoder. The ratcheting process uses the same reinitialization procedure as the Star-Generated Model (SGM), however, ratcheting can only be performed once per model with each subsequent ratchet being a descendent of the initial model. In this way, a ratcheted model does not have any "siblings" that were generated from the same "parent" model.

Star-generated Model One of a number of steganographic models created by reinitializing the final layers of the ML model encoder and decoder. The specific generation mechanism uses the same "base" model producing a "star" shape with the base model in the middle and the new models as the points of the star. Another way of understanding the model is that the base model is the "parent" model producing many "child" models whom are all "siblings."

Unique Decodability Unique decodability refers to stego-images produced by each individually trained model only being decodable by that specific model. In other words, stego-images produced by one model cannot be successfully decoded by another.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank my thesis advisors, Professors Dr. Britta Hale and Dr. John "Vinnie" Monaco. They both strongly encouraged me to pursue research topics that, in addition to being relevant to National Security and the interests of the United States government, aligned closely with my knowledge, skills, and interests. The material covered in classes they each taught was a major source of inspiration for this research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Barriers to communication exist in various forms, such as physical distance, differences in technology, and local rules and regulations. Although modern telecommunication's infrastructure and the Internet have reduced the impact of physical distance and lack of technology, local rules and regulations regarding how people communicate with one another remain a significant communication barrier for many citizens around the world. Our research outlines the issue of digital censorship, particularly within authoritarian regimes. We discuss current commonly used methods to subvert censorship and evaluate them. Finally, we propose, test, and evaluate a Machine Learning (ML) steganographic solution to subvert digital censorship for use alongside existing messaging applications. This chapter outlines the main premise, objectives, benefits, and contributions of our research.

1.1 Problem Definition

Digital censorship is commonly used in authoritarian regimes around the world [1]. Even countries that have been historically regarded as having a "free" internet, such as Norway, have shown increased censorship activity in 2020 [1]. However, the greatest level of censorship occurs in Russia, Iran, and China [1]. Given China's recent and economic and military rise to a peer or near-peer competitor of the United States, there is great concern that China will export not only its surveillance technology but also its worldview in regard to restricting internet freedom. Due to the Chinese government's efforts to prevent, control, and quell social unrest, it has developed a robust and powerful infrastructure for domestic surveillance and censorship. As a result, communications via Chinese-developed messaging applications and Chinese domestic infrastructure are particularly vulnerable to surveillance and censorship. Researchers have shown that even international users of WeChat operating from outside of China unwittingly participate in the surveillance and censorship mechanisms of the application [2]. International user communications inform the domestic image and content blacklists thereby enhancing the application's domestic censorship capabilities [2]. A high degree of surveillance is especially concerning for dissidents, social organizers, foreign citizens, and non-Han ethnic groups, such as the Uighurs. Therefore we propose an

adaptive image-based steganographic model for use with censored messaging applications to circumvent censorship in support of a free and open internet.

1.2 Objectives

The main objective of this research is to develop and evaluate a model for surveillance and censorship subversion that can be implemented with existing messaging applications. The goal is to provide a covert, secure mechanism for sending embedded messages via an existing communication medium, thereby preserving usability and social networks and also enabling the further spread of this technology. Given the nature of and general attitude towards digital censorship in China, our model will be geared toward providing small-scale, long-term undetectability for a small number of users. That is to say the goal is to protect directed communications over the long-term vice broadcast communications. Based on our research, the demand for a privacy-enabled, censorship-circumventing application in China is only lukewarm [3].

However, those that do want access to such technology are at significant risk from authorities and thus long-term secrecy is a desirable feature. Even if the use of steganography is discovered, message content should be protected via cryptography. At the time of this writing there exists no viable anti-surveillance, anti-censorship messaging application available in China [4]. Signal is one of the most recently banned applications. The Signal application's protocol has strong security guarantees and is utilized in some of the most popular messaging applications, including WhatsApp and Facebook Messenger [5]. However, historically, many Western-developed applications, especially those with strong security guarantees, eventually are banned by the Chinese government.

It is believed that WeChat is so popular and entrenched in the lives of Chinese citizens that it would be extremely difficult for the government to ban its use outright. For example, WeChat functionality goes far beyond that of Facebook and WhatsApp and integrates messaging, social media, digital payments, medical and official business, and educational functions [6]. In other words, it can be thought of as an amalgamation of Facebook, Blackboard, Sakai, Office365, and Venmo, and probably other applications. For the reason that the application is so popular and has essentially become indispensable, we choose to develop our model with the WeChat platform as a backbone with a steganographic framework on top as a

means to subvert surveillance and censorship. A steganographic model seeks to hide secret messages within the noise of normal communications and this platform provides plenty of noise within which to hide. Although this model could theoretically be applied to any messaging application, it was developed with the intention of being used in countries that lack free speech protections and have intrusive surveillance and censorship practices, such as China. However, Chinese citizens are not the only people at risk from corporate and government snooping. The steganographic model presented here is meant to be flexible and support covert communication over any insecure communications channel.

1.3 Benefits of Study

1.3.1 Current Methods of Censorship Circumvention

Although circumventing censorship in China is difficult, there are several options available. Many internet users in China are familiar with VPNs [3]. However, VPN use is restricted by the government to certain providers, whom the government presumably has access to their data. Only users savvy enough to download and use a non-government approved VPN have a chance of their activity not being monitored by the government. However, those users have virtually no guarantee that they are not being snooped upon as even supposedly secure VPNs are not invulnerable to advanced nation-state spying. Besides using VPNs, citizens in China have few options to circumvent censorship other than meeting in person. Of course, speaking or writing in some sort of code is an option, but this is only a temporary measure as authorities usually catch on fairly quickly.

In Chinese internet slang, there are many words that are substituted for near homophones to convey some sort of additional, hidden meaning. For example, the Chinese word 河蟹 (héxiè; river crab) is a near homophone for 和谐 (héxié; harmony) differing only in tone (see tone marks) [7]. Government censors can quickly catch on to attempts to mock official language but are slower to block seemingly innocuous terms, such as river crab. However, much of the use of these kind of near homophones has been used in a mocking or joking context and quickly becomes obvious to government censors. However, at its roots, this kind of wordplay is a very rudimentary form of steganography. You replace a sensitive word with an innocuous one with similar pronunciation and the likelihood of being censored decreases. The obvious drawback of this approach is that it is very easy to figure out. Once

the connections between homophones are understood, there is no confidentiality and past messages are vulnerable to censoring. Any modern stego-system would be an improvement in regard to secrecy.

Besides VPNs and internet slang, there are few other techniques to subvert censorship. A robust steganographic system that utilizes cryptography would combine clandestine communication from steganography with all the security guarantees of encryption.

1.3.2 Impact

Any government, organization, or individual that values free speech will benefit from a means to subvert censorship. From citizens who are casually interested in sharing political "memes" to those who are determined to exercise collective action in support of political reforms, workers' rights, or even regime change, potential users can utilize existing applications and social networks with very little overhead. These communications would not only be protected from censorship but also from surveillance as they appear innocuous, holding a key advantage over encryption. Since those most interested in this technology are likely to already be under strict government surveillance, the last point is of even greater importance.

In regard to benefiting the U.S. government, the recently declassified U.S. Strategic Framework for the Indo-Pacific includes the objective of enhancing U.S. engagement in the Indo-Pacific region while educating local populations about the Chinese Communist Party (CCP)'s "coercive behavior and influence operations" [8]. As a means to accomplish this objective, the document contains the following specific action: "invest in capabilities [redacted] that promote uncensored communication between Chinese people" [8]. Our research involves building a steganographic model for use with existing Chinese messaging platforms and thus directly supports the guiding strategy for the executive branch and the National Security Strategy. If our research is successfully implemented, it will serve as a secure means for citizens around the world to communicate with individuals within China, or in other countries with repressive regimes, without arousing suspicion. Although the U.S. government undoubtedly already has the means to communicate with those inside China and is making investments elsewhere, this research would be yet another channel, potentially enhancing operational resiliency. Operational command and control and/or support for elements operating inside China is much safer if accomplished remotely. Adaptive

steganography might be yet another tool when encrypted communications are not available but secrecy is required. Additionally, the potential impact of this model is greater than just affecting citizens living under repressive regimes. The model presented here is something of a steganographic "band-aid" that can be placed on existing messaging applications and provide a secure covert channel. Other than the download of the overlay/plugin, the use of this model is meant to undetectable by design, which will we later test and evaluate.

Of course, some critics would be quick to point out that this kind of technology has potential to be abused. Recent reporting has shown that many of those who participated in the riots that occurred at the Capitol building on January 6, 2021, coordinated their actions via the messaging application Parler [9]. Due to the poor implementation of the application, user data collected by the application was easily obtained and turned over to authorities [10]. These events serve as a cautionary tale to would-be pro-privacy application developers and has two main lessons. First, tools designed for evading censorship and surveillance can be used for nefarious purposes. Although the term "nefarious" can mean different things to different people, in this case, the platform was allegedly used for spreading anti-democratic sentiment and ultimately appears to have been used to coordinate an insurrection against the United States government [11]. Therefore, this kind of technology can be thought of as a double-edged sword that can accomplish the very opposite of what was intended. For example, historically, the United States has consistently championed the principle of democracy in an unqualified fashion. However, the Muslim Brotherhood in Egypt is arguably not a political force that the U.S. government would prefer to be in power: a true commitment to ideology requires a compromise in terms of pragmatism.

The other major lesson to be gleaned from Parler's role in the Capitol building violence was the fact that private user information was not protected by the application designers and then leaked to authorities by third parties. Although it would be difficult to imagine this as anything but a good thing, meaning that those involved in illegal behavior will be held accountable for their actions, imagine a similar situation except occurring in China. Imagine instead that pro-democracy activists in Hong Kong were using a similar application that championed anti-censorship values. Then also imagine user data was hacked and leaked to Chinese authorities leading to hundreds of arrests and disappearances. The key point here is that poor technological implementation can actually make an "anti-censorship" app into a sort of honeypot, only attracting the sorts of individuals that the CCP would be

most concerned about. One could imagine a scenario in which the CCP designs a so-called pro-privacy, anti-censorship app with the ultimate goal being to root out any potential troublemakers in the population. There is Chinese historical precedent for such behavior, such as Mao Zedong's Hundred Flowers Campaign in which Mao Zedong invited criticism from non-party affiliated intellectuals only later to retaliate against those same individuals, sending many to labor camps and prison [12]. Therefore, serious attention should be paid to how anti-censorship software will be implemented and disseminated as many Chinese will likely be wary of potential backlash from the CCP.

However, it is important not to fall into the logical fallacy of assuming that just because someone is using encryption, steganography, or other privacy or anti-censorship enabling technology that they are in fact guilty of some kind of wrongdoing. For example, the widespread use of encryption has been hotly debated, both in the U.S. and internationally, the last several years due to the fear that criminals and terrorists will utilize the same tools and techniques to protect their privacy and enable illegal behavior. In the case of China, the government's answer to the question of whether or not to allow privacy-enabling encryption was to ban any software that the government cannot directly monitor. The answer to whether the U.S. and others should allow encryption and steganography to be freely used follows directly from whether citizens, society, and a nation believe that all individuals have the right to privacy and free speech. In the case of China, the government does not believe an individual's right to privacy or free speech ever outweighs the good of the collective group or the potential risk to state security [13]. In this context, the good of the collective group specifically refers to the CCP's interpretation of what is best for all Chinese citizens. As for the United States, per the 2017 National Security Strategy and the 2021 Interim National Security Strategy, it is in our country's best interest to spread and promote liberal, democratic values and champion free speech [14]. There may be no better way to do that than to create and export a platform where it can be accomplished at low risk and easily accessed by average citizens.

1.4 Contributions

This research makes the following specific contributions to creating a capability that enables and promotes the uncensored communication between people:

- Demonstrates a technique to ratchet a ML-generated steganography algorithm.
- Tests and evaluates the efficacy of increasing steganographic embedding capacity through a tiling process.
- Develops and evaluates a holistic framework and communications protocol for circumventing censorship utilizing steganography and cryptography.
- Evaluates a modified steganographic algorithm against steganalysis techniques.
- Designs and evaluates an effective statistical steganalytic attack to identify ML steganography images based on distribution of decoding bit-errors.
- Identifies key shortcomings and challenges to design such a communication model.

1.5 Thesis Structure

In the following chapter, we will provide an overview of key topics and terms such as privacy, censorship, and steganography in addition to a summary of the history and current status of Chinese digital surveillance and censorship. In Chapter 3, we will discuss the threat model we use to inform our overall steganographic model's requirements. We then specify our model's requirements and explain how we selected an existing model to modify in accordance with our needs. In Chapter 4, we explore the results of numerous experiments designed to evaluate our model's ability to to satisfy our requirements for an overall steganographic messaging solution. Finally, in Chapter 5 we examine key conclusions drawn from results of the experiments discussed in Chapter 4 in addition to key challenges and potential vulnerabilities to fully developing and implementing our steganographic model.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

In this chapter, we discuss key concepts and background material that provide context for our research. We first explore the relationship between privacy and censorship and its relevance to digital communications policy and technology. As steganography and cryptography are common tools used to subvert censorship, we provide a basic overview of important concepts related to steganography, steganalysis, and cryptography. We then discuss some specific digital steganography and steganalysis techniques. Finally, we explore the history and status quo of Chinese digital surveillance and censorship before we introduce the main goal of our research: to develop an adaptive, ML steganographic solution to effectively subvert censorship.

2.1 Privacy vs. Censorship

Before we discuss the general issue of censorship, it is important to define two key terms and explain their relationship. Censorship is removing content deemed harmful to a population. A key question is then who decides what is deemed harmful? The main debate surrounding censorship typically revolves around who decides what is harmful. Many countries and populations have different answers to this question, thus producing the variety of censorship regimes present in the world today. Censorship enforcement includes removing selections from media, such as parts of books, movies, or letters, or banning the sharing, selling, or distribution of certain material outright. Claimed goals of censorship are often noble. Governments that regularly engage in censorship often claim that they are doing so for the benefit of their people. Some argue that governments censor material that they deem threatening to themselves.

Although many in the United States support freedom of expression, which in many ways runs counter to the principle of censorship, censorship remains commonplace. There are many historical examples of censorship of American films, books, and artwork [15]. However, censorship, regardless of the justification, is in conflict with the very American notion of free speech. American laws regarding censorship are vague and poorly defined. For example,

readers may be familiar with the phrase "I know it when I see it," which is a reference to Justice Potter Stewart's attempt to define "hard-core pornography" during the 1964 Supreme Court case Jacobellis v. Ohio [16]. In summary, freedom of expression and censorship exist on a spectrum with one extreme being complete and utter freedom of expression and the other, with the government completely controlling the speech and even thoughts of citizens by only allowing them to express themselves in government-sanctioned ways.

Although this research aims to develop a generalized technical solution to enable citizens living in countries whose governments' policies tend toward the latter description, it is first important to understand how such an extreme level of control is possible. In order to build and maintain a robust censorship capability, a government must first be able to access, exploit, and potentially deny any communication channel it wishes to censor. In other words, censorship, especially at an individual level, can only exist if it is buttressed by a massive surveillance infrastructure. For citizens living under such restrictions, it is difficult to imagine individuals maintain any level of privacy. For our purposes, we will define privacy as freedom from observation, either by the government, corporations, or other citizens. The "surveillance state" is inherently at odds with the principle of privacy. Therefore, in order to practice censorship, some level of privacy needs to be violated. This relationship can also work in reverse: if one can gain true privacy, he or she cannot be censored because, by definition, the censors will not be able to observe the communication taking place.

Consider the common example of Alice and Bob as two parties communicating. Government spies fill the role of Eve, who is passively intercepting Alice and Bob's communications. As Bob and Alice exchange messages back and forth, Bob sends a message considered by the government to be "sensitive" or "harmful." As soon as Eve detects the potentially "harmful" message, she then goes beyond eavesdropping and plays the role of an active attacker, who deletes the content and/or blocks the communications channel entirely. Our research explores the possibility of creating content that, although it can be easily observed by Eve or Mallory, it will never appear harmful and thus never be censored. This can be accomplished by creating a message within a message that can only be observed if one knows exactly where to look, also known as steganography.

Steganography is simply hiding a secret message within a "cover" message such that an

outside observer, even if they see the cover message, does not perceive an embedded secret message. The history of steganography goes back to the time of Ancient Greece, with the term itself translating to "covered writing" [17]. In modern times, it was used in both the World Wars to pass secret messages written in disappearing ink [17]. Messages written with disappearing or invisible ink would only become visible when a certain chemical was later applied to the dried ink [17]. Another modern example of steganography is the microdot. A microdot is a miniaturized version of a text or image, sometimes so small that they resemble a dot, hence the name. These kinds of technologies were used by spies as far back as the Franco-Prussian war, during both World Wars and likely up through today [17]. These forms of steganography rely upon technical solutions to obscure a hidden message, but once the means of the hiding is discovered, the secret message can easily be seen as it is not encoded in any meaningful way.

A related form of steganography that can complicate the decoding process is linguistic steganography that uses codes or codewords to symbolize a second meaning. Anyone who has watched the first season of the popular HBO Show *The Wire* has seen an example of this kind of steganography. Without providing too many spoilers, the plot revolves around a police strategy to impede the activities of Baltimore drug dealers, primarily through intercepting communications [18]. The drug dealers quickly learn to avoid certain communications methods and invent codes of their own. At first, they adapt by obscuring phone numbers sent to pagers by flipping the original number across the axes of a phone keypad, or in the words of the characters, "jumping the 5" [18]. For example, if they wanted to send the number 1234, they would send 9876. In later seasons, the steganographic system transformed into image texting where exchanged images were of a clock face with the clock hands corresponding to specific locations in the Baltimore area. Eventually the police caught onto both steganographic methods, which, as we will discuss later in the paper, is indicative of the generally weak security guarantees provided by steganography. Although the techniques discussed above utilize digital communications infrastructure, the underlying mechanism simply maps clock hands to numbers corresponding to locations. Digital steganography involves actually modifying bit values of the data being sent. These techniques create the possibility for embedding much more data than analog techniques and making discovery of the steganography much more difficult (see Section 2.1.3 for more details).

2.1.1 Steganography vs. Cryptography

Generally, steganography is hiding data within other data. The cover, which is the data within which a message is hidden, can be in the form of text, an image, or any kind of digital file. The goal of steganography is to hide a secret message within a cover to hide the fact that such a message even exists. A more technical way of describing steganography is that it makes use of covert channels within a given medium to secretly relay data. In digital image steganography, the bits in a given cover image must be modified in such a way that any changes are both visually imperceptible and difficult to detect by any other means.

Steganalysis is the process of detecting the underlying encoded stego-messages, or the output messages produced by steganography. In general, steganography does not have the same types of provable security guarantees, such as Confidentiality, Integrity and Authenticity, that commonly used cryptographic algorithms such as Advanced Encryption Standard (AES) do, which we will explain below. However, the main appeal of steganography over cryptography is its covert nature. In other words, effective steganography is clandestine while effective cryptography is not required to be. Encrypted data, otherwise known as ciphertext, is typically easily distinguishable from unencrypted data, or plaintext. For example, the visual representation of an encrypted image is similar to white noise and displays high entropy. Likewise, an encrypted text file contains random-looking (and should be quite close to random) bytes. In other words, encrypted files are not understandable and are clearly different from non-encrypted files. On the other hand, a stego-image, which is a cover image embedded with a secret message, appears as a normal image with no obvious distortions or artifacts.

Although the main focus of this thesis is steganography, the scope of research includes cryptography and several related principles, such as key generation and ratcheting algorithms. The most fundamental principles on the use of cryptography were developed by Dr. August Kerckhoff and published in 1883 [19]. He laid out six total principles for a secure telegraphic system. Today, Kerckhoff's Principle is most closely associated with just one of his six principles: the system must not require secrecy and can be stolen by the enemy without causing trouble. Although nearly all modern cryptographic applications adhere to this principle, steganography does not have the same security guarantees. In fact, steganography relies on "security through obscurity," which is clearly at odds with Kerckhoff's Principle. For example, one reason for AES being considered secure is that, statistically, there are

so many possible solutions for the shared private key that it is computationally infeasible to try all keys. As for the algorithm itself, it is an open standard and was created with significant input from the global cryptographic community [20]. Over the past twenty years, many researchers have attempted various attacks against the algorithm which have not been successful [21]. Given that the algorithm standard is open, as long as the key is protected, the secrecy of the system itself is unimportant. Conversely, most pure steganography algorithms, those not incorporating some form of cryptography, are compromised as soon as the algorithm is guessed or identified.

In the case of AES, just because an attacker knows that a message was encrypted using AES does not mean they will be able to decrypt the message. Assuming the algorithm is secure and has no known attacks, which the National Institute of Standards and Technology (NIST) suggests is the case [21], the attacker needs to know both that AES was used and the shared key. If one's main goal is to prevent others from reading a secret message, then cryptography is the best solution. However, if the goal is to both hide the fact that a secret message is sent and protect the contents of that message, then both steganography and cryptography can be used together. A simple implementation is described below: first, a message is encrypted, then encoded into a cover image or document, sent over some communications channel, and then the message will decoded from the stego image and then decrypted. This process requires both sender and receive to use the same steganographic and cryptographic algorithms and keys. We assume a similar premise of use for this work.

What makes steganography particularly vulnerable is that, in many cases, the system itself is the secret. For example, if Alice and Bob are using microdots to hide their messages, a would-be attacker need only know that they are using microdots and they can easily acquire a magnifying glass and decode the messages. A more robust implementation of steganography would include the flexibility of cryptography-like keys which enable each pair of users to have a unique key which prevents unauthorized viewing by other parties. In the case of ML-generated steganographic models, the weights of the models might be thought of as a sort of key. In order to encode and decode a given cover image and stego-image respectively, both sender and recipient will need to use the same model, or in other words, the same neural network structure with the same weight values. The uniqueness created by these different weights makes the attacker's job more difficult by requiring that they know not just the general model used, but also the specific weights, which depending on the size of the

"key space" or total possible values of weights that result in a usable model, could introduce significant computational overhead (see Section 5.2.5 for further discussion of exploring the key space of our model). However possible collisions and viable distance required between weights for usable models must be assessed to determine the "key space." Our research explores the possibility of using neural net weights as steganographic "keys" to strengthen the minimal security guarantees associated with most applications of steganography. Note also that by this type of interpretation of "keys," where the network itself may be known without compromising security, we can align the steganographic technique to Kerckhoff's Principle. Nevertheless, we acknowledge the limitations of steganographic security and assume the steganographic implementation is used in tandem with cryptography. From this point on, discussion of steganography will be focused on ML-based steganography techniques.

2.1.2 Steganalysis Basics

The simplest definition of steganalysis is the process of identifying the use of steganography [22]. However, steganalysis can also refer to process of discovering the original secret message. Identifying the use of steganography can be accomplished with a binary classifier or assigning a given image a probability value of containing steganography. On the other hand, message recovery is much more difficult. Just because a given steganalysis technique can accurately determine whether an image is a stego-image, which is an image that contains a secret message embedded by some steganographic algorithm, it does not mean that technique will be able to recover the original secret message [22]. For example, one can employ a passive steganalysis method of processing an image by flipping every Least Significant Bit (LSB) per pixel in an image to destroy any potential LSB-encoded secret message without significantly altering the perceptual quality of the image [23]. Although the secret message will have been destroyed, it will not recovered by the attacker or the intended recipient.

A stego-image's ability to withstand image processing and compression techniques without corrupting the message can be referred to as a steganographic technique's robustness. In other words, robustness is a measure of a given steganographic algorithm's resistance to image processing and compression techniques [23]. A less robust technique decreases the chances that an embedded secret message will reach its intended recipient, however,

it also reduces the adversary's chance of exploiting the secret message. Depending on the implementation, robustness may or may not be a desirable quality. In watermarking, robustness is a key requirement as a watermark functions as a permanent label or signature that should be "undeletable." In steganography, where covert communication is the goal, it may be desirable to choose a non-robust technique that "erases" an embedded message upon any image processing or compression. In this way, an extremely fragile (non-robust) steganographic technique could be used to perform fuzzing of image processing techniques. The recipient could analyze the received image and determine the level of image processing that had been performed on that image and thus tailor a steganographic solution to avoid techniques that are vulnerable to the identified methods of image processing used.

2.1.3 Digital Steganography and Steganalysis Techniques

There are many different digital steganography and steganalysis techniques. The simplest kind of digital steganography takes advantage of file formats. A file contains an End Of File (EOF) tag, which marks the end of the file. However, one can still write data beyond this marker. When a JPEG file which has information added beyond the EOF tag is viewed in a photo viewing application, there is nothing different about the image to suggest that extra information has been added precisely because that information is not present in the displayed image [23]. However, if one were to examine the file in raw hexadecimal notation, the added information would be clearly visible. Techniques such as this are quite easy to detect and have virtually no security guarantees. Many other types of digital image steganography exist in the spatial domain involving the modifications of pixel values in a given cover image [23].

Any successful steganographic method must be resistant to common basic steganalysis techniques. Techniques include visual steganalysis, structural steganalysis, statistical steganalysis, and learning steganalysis [22]. If sent images are being manually monitored for sensitive content, then the steganography must not be visually apparent; in other words, it must be immune to the most rudimentary form of visual steganalysis. Visual steganalysis can also involve examining transformations [22] of an image to look for any visual anomalies. Structural steganalysis requires examination of the file format or structure of a file for any signs of modification [22]. Statistical steganalysis typically involves making a judgment whether an image is a stego-image based on if certain parameters of the image fall within a theoretical probability distribution of a stego-image generated by a certain

stego-system [22]. Learning steganalysis is nothing more than a steganalysis technique that draws on those previously listed but enabled by machine learning. It is likely that emerging and future censorship techniques will take advantage of technological achievements in both ML and Artificial Intelligence (AI) to detect censored images and the use of steganography to hide content.

LSB is a common technique that involves replacing the LSB pixels in an image with the binary corresponding to a secret message. The recipient of an LSB-encoded image need only to read the LSB of each pixel to find the hidden message. Although more advanced than just adding extra information in an unexpected location within a certain file format, this kind of steganography is vulnerable to statistical steganalysis techniques such as the Chi-squared and sample pairs analysis (SPA) attacks. Chi-squared, Regular-Singular (RS), Weighted-Steganalysis (WS), Sample Pairs Analysis (SPA) and Difference Image Histogram (DIH) are common statistical analysis techniques [24]. These techniques are also known as structural attacks that look for irregularities in the spatial domain of images. Other state-of-the-art spatial domain steganography algorithms include Highly Undetectable Steganography (HUGO), Wavelet Obtained Weights (WOW) and Universal Wavelet Relative Distortion (UNIWARD) [25]. These techniques all differ in the precise way in which they minimize their own distortion metric, which implies they will be vulnerable/resistant to different types of steganalysis [25].

In addition to steganography algorithms that operate in the spatial domain, there are those that modify the frequency domain. These algorithms apply a mathematical transformation, such as Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT), and Discrete Transform (DFT), to the cover image resulting in changes to a number of coefficients that correspond to a block of pixels [23]. These coefficients can be carefully modified to embed a secret message without causing any noticeable alteration to the original image. Although algorithms implemented in the frequency domain may be slightly more complicated than those in the spatial domain, they are susceptible to the same kind of statistical attacks.

In order to resist steganalysis, our model will be adaptive in exactly how a secret message is embedded in an image file. The term “adaptive” refers specifically to the weight values in the neural net of our model. These weight values are the “key” to how bits are embedded and thus knowing the exact weight values is essential for decoding the message from a

stego-image. An adaptive model will be less susceptible to targeted steganalysis techniques developed to detect usage of known stego-systems [22]. Research has shown that certain steganographic techniques, such as LSB where a secret message is embedded in the least significant bits of an image's pixels, can be reliably detected with statistical steganalysis [25]. Additionally, other well-known steganographic techniques such as HUGO, WOW and S-Uniward can be more easily detected than some machine learning generated models such as HiDDeN [25]. By using a novel algorithm or a modified, adaptive version of an existing algorithm we can increase a potential adversary's computational overhead by requiring them to run multiple different targeted steganalyzers [22]. However, a model which guarantees long-term security must also be immune to future steganalytic attacks. Recent research has identified the potential for using adversarial training to develop both steganographic models and steganalyzers that compete with existing models in terms of visual perceptibility and steganography detection [26]. It is likely that machine learning will be used to improve upon existing steganographic systems and to develop better performing steganalysis. Artificial Training Sets (ATS) is an example of a current state-of-the-art ML-based steganalyzer that was developed from machine learning [27]. It relies upon training a steganalyzer on certain algorithms and embedding rates. Thus, our model must not only provide protection against current state-of-the-art ML-based steganalysis techniques such as ATS, but also against algorithms produced by unsupervised learning [26].

2.2 Chinese Digital Surveillance and Censorship

Over twenty years ago, then President Clinton proclaimed that China's attempts to "crack down on the internet" were like "nailing jello to a wall" [28]. However, those efforts seem to have largely succeeded. Not only has the Chinese government succeeded in filtering its internet but it did so without radically upsetting its populace while simultaneously experiencing a major economic and technological boom. Although most Chinese are aware of the invasive nature of their government's surveillance apparatus, they hold complex attitudes toward censorship. Many Chinese are indifferent or even generally supportive of censorship as they believe it prevents the spread of misinformation [3]. What many in the West would see as a gross overreach of government power and a clear violation of free speech, many Chinese see as a necessary evil that protects people from themselves. However complex Chinese attitudes towards censorship may be, there exist certain elements

of the population who wish to share sensitive information covertly. Protests in China are becoming more common in recent years and there have been consistent, usually localized movements in support of issues such as worker's rights [29], the environment [29] and anti-corruption [29].

Although subverting censorship may not be of interest to the average Chinese citizen who is just trying to keep their head down in their quest for the "Chinese dream," labor organizers, protest leaders, ethnic minorities, and NGOs, or anyone else seeking to share sensitive information require a suitable platform. It is important to note that these concerns are not only harbored by militant, anti-government dissidents, but by some ordinary citizens who are concerned with being flagged by authorities. For example, how does one ask a family member in China how the COVID-19 situation is locally? The reason behind the question is concern for one's family; however, the answer may well be considered sensitive by the Chinese government. Leaked documents from the Cyberspace Administration of China demonstrated how the government heavily censored sensitive COVID-19-related events, such as the death of Dr. Li Wenliang, over many media platforms [30]. Anti-censorship mechanisms can enable family members to communicate with each other across vast distances, regardless of the "sensitivity" of the topics. Unfortunately, in China, there are very few messaging applications allowed for download and/or use within the country and even fewer that have any reasonable security guarantees.[1] As a result, Chinese government surveillance of messaging applications is simplified because nearly all messaging traffic is on a few select, government-approved platforms. Most users are relegated to using Chinese-developed software which is routinely monitored and censored by the government [31]. Although digital censorship across Chinese communications platforms is both ubiquitous and continuous, current detection techniques of targeted content are largely rule-based [32].

Billions of people, both in China and beyond, utilize Chinese-developed messaging applications: WeChat itself has over a billion global users [32]. All messaging applications available for download in China, to some degree, practice censorship, and by extension surveillance, as required by Chinese law [32]. Research has identified Chinese-developed messaging applications mainly utilize censorship models centered around flexible keyword and image block-lists to both preemptively block and retroactively remove sensitive con-

[1] At the time of the writing of this paper, the messaging application Signal is still usable in mainland China; however, it is not available for download from the Apple or Android digital stores.

tent [32]. In many of these models, posts and images can be automatically flagged and then later manually reviewed [32]. Even if users were somehow able to use a VPN, which some tech-savvy Chinese netizens do, the use of encryption could be easily identified and may cause flagging of certain users for increased surveillance and monitoring [3]. To summarize, there are virtually no secure messaging applications available for use in China. Additionally, switching to a new messaging app requires others to switch to the same app to gain any security benefits.

Researchers have discovered that WeChat uses a combination of automated Optical Character Recognition (OCR) and statistical techniques to identify and censor sensitive images in near real-time [32]. Perhaps the most widely known example of an image commonly censored by Chinese authorities, and considered by some to be one of the most recognized photographs of the 20th century, is the 1989 photograph of "Tank Man" in Tiananmen square [33]. It consists of a man in a briefcase standing in the path of three tanks in Tiananmen square in Beijing and is considered symbolic of the protests and modern-day resistance to authoritarianism [33]. In order to evaluate whether an image is considered sensitive by WeChat, it goes through a number of checks. The first check of an image is performed client-side as the image in question is compared against a hash index of MD5 hashes of known, blacklisted images [32]. As such, flipping the orientation of a known, censored image upside down is not enough to circumvent censorship. The second check occurs after the image has been sent and is checked via OCR, statistical, and manual visual techniques for sensitive content. This two-stage filtering process utilizes the hash check to minimize computationally expensive OCR and statistical checks when the image in question has already been identified. However, the second stage does not occur instantly and blocked images can remain visible for some time. Any image whose hash does not exist in the blacklist index and does not appear to contain sensitive content will not be blocked. Therefore, if the researcher's main conclusions are correct, even simple steganography will not be detected.

Even if there is no immediate detection, there still exists the risk of after-the-fact detection which could result in consequences ranging from deleting the original content, to banning accounts, and even potential legal consequences. Therefore, simple steganography techniques are not adequate anti-censorship tools because they merely reduce the likelihood of short-term censorship and depending on the message content, potentially increase the long-term risk to all parties involved by providing a history of exchanged messages through

which censors can sort, should the steganography later be discovered. Additionally, although current filtering techniques are somewhat rudimentary, these techniques will only improve over time as they take advantage of technological improvements in ML and AI. However, although Chinese authorities have an expansive workforce enforcing censorship, the sheer number of exchanged images precludes manual analysis of every image shared across all messaging applications. As a result, a significant portion of current image censorship is automated. Although this kind of image filtering generally requires massive amounts of resources, the Chinese government has demonstrated its incredible ability to nail jello to the wall time and again. In short, the Chinese government will only improve its messaging censorship capabilities and currently available anti-censorship techniques will become less effective over time.

We therefore propose an adaptive, image-based steganographic model for use with popular messaging applications that avoids automatic censorship mechanisms, provides cryptographic protections, and is visibly imperceptible. Such a model would enable users to communicate sensitive information clandestinely and with strong security guarantees, which is a capability that is otherwise non-existent in mainland China.

CHAPTER 3:
Threat Model and Model Requirements

This chapter introduces the threat model for censorship in messaging applications and the requirements for operation of a successful censorship subversion model. First, we discuss the basic motives, capabilities, and constraints of the threat actor. Next, we discuss countermeasures to overcome the threat actor's censorship capabilities in the form of our model requirements.

3.1 Threat Model

The threat model that our overall communication framework was created around is based on the Chinese censorship system. More generally, we assume the threat to be a well-financed, technologically advanced, and highly motivated and capable nation-state. Therefore, in the development of our framework, we assume that the adversary has the means and capability to passively access all local and cloud-based information for any user. Additionally, the adversary can be assumed to have full control of the network (e.g., the adversary has ability to replay, delete, delay, or inject messages). It also has ability to control devices on the network, but not the device endpoints (i.e., controlling device C should not affect the adversary's ability to read A's and B's messages to each other). We acknowledge it is possible that many authoritarian governments may have the capability to selectively access and control a target end user device at a small-scale, but as our model is aimed at circumventing mass censorship, these techniques are not within our threat model and fall outside the scope of this thesis. Many other governments around the world utilize similar technology-based censorship methods and China, although it is arguable the most proficient is merely a case example.

Not only do we assume that the adversary is capable of intercepting and manipulating internet protocol and message traffic, but that they are also capable of filtering and analyzing the data. Although we discussed some low-tech methods currently employed by censors working for WeChat, we assume the adversary has a much greater potential capability to identify any sort of content deemed malicious, which could include anti-government messaging,

pornography, encryption, or steganography. Content deemed malicious extends beyond the aforementioned categories to content that "spreads rumors or disturbs social order," "insults or defames third parties," or "jeopardizes the nation's unity" [34]. We assume that such passive identification and active filtering of network traffic can be accomplished if a given user is already flagged or targeted. Additionally, it is assumed that real user identification is required to register accounts, as is the case for most platforms in China [34]. If advanced capabilities are not used, we assume that the adversary will develop these capabilities to identify malicious or inappropriate content, likely taking advantage of advances and machine-learning.

3.2 Model Requirements

3.2.1 Visual Imperceptibility

Although this requirement seems straightforward, proving visual imperceptibility is troublesome. Research has shown that commonly used metrics for measuring visual perceptibility such as Euclidean distance, Pseudorandom Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are not always good approximations for what is detectable by human vision [35]. Proving humans cannot visually discern differences between two images is impossible without a concrete standard, which does not appear to exist. For our purposes, visual imperceptibility means a human observer would not be able to detect any obvious signs of image manipulation. This definition is purposefully vague as it provides us some flexibility to determine what is visually imperceptible. Given the variety of the kinds of images people share, there is significant room for the use of filters, image processing or other photo editing techniques. However, the manipulation must not be easily identifiable as some kind of inadvertent remnant of steganographic manipulation, such as strange coloration or pixelation in parts of the image. Additionally, even non-altered images can appear to have some kind of odd mark as a result of lighting or particulates in the air. Visual perceptibility can be assessed through surveys of test subjects. This introduces other potential issues such as human error and how best to frame the test to minimize the possibility of test subjects imagining differences where none exist. Such assessments are left to future work, and we will aim to address visual imperceptibility only as far as very noticeable changes in a cover-image and instead focus on other forms of steganalysis. Although this method is somewhat

unscientific, the underlying goal is to "hide within the noise" of other shared images. We will include stego-images in the paper and let readers themselves evaluate whether or not the steganography is visually imperceptible.

3.2.2 Resistant to Steganalysis

Our model must be resistant to basic statistical steganalysis techniques as well as more advanced ML-enabled techniques. Images generated by our model will have to stand-up to the best known steganalysis techniques with the goal of correct classification being "50 percent": any deviation from 50 percent implies the model can detect steganography with better than random chance. Our model is primarily concerned with overcoming blind steganalysis which is steganalysis that is not targeted at detecting the use of any one particular steganographic technique [22]. However, we will also attempt to create a model resistant to targeted steganalysis. If a particular type of steganography becomes popular enough, nation states will eventually recognize this and tailor steganalysis to counter the most commonly used algorithms.

3.2.3 Long-Term Security

The simplest way to guarantee the long-term security of a message is to use provably secure encryption. Before embedding the user's designated secret message into a cover-image, the message will be encrypted, such as the Signal protocol. The cryptographic Signal protocol has security guarantees beyond most messaging platforms, particularly forward secrecy [36]. This is accomplished by a ratcheting mechanism in which a new encryption key is generated from several inputs supplied by each user. The term "ratchet" is used because, like the tool, it can only operate in one direction, creating a new key that is totally incompatible with previously generated keys. Our research involves answering the question of whether a machine-learning based steganographic model can be ratcheted in a similar way.

3.2.4 Unique Solutions Per Pair of Users

This requirement incorporates steganalysis resistance and long-term security. In current uses of steganography, such as LSB, the ability to decode the encoded message requires only that the recipient know the algorithm that was used. In LSB steganography, the least

significant bit of each pixel is read and those values are put together to reconstruct the encoded message. The steganographic "key" is the algorithm itself.

However, in machine-learning generated steganography, the "key" comes in the form of the structure of the model and its particular weights. In order to decode ML-generated steganography, one must know the general algorithm (the structure of the neural network) and the weights associated with the layers of the network(s). We require our model be able to generate a unique steganographic solution per pair of users in a way that does not negatively affect usability. Specifically, we wish to avoid collisions between models used by different pairs of users. A collision can be defined as a degree of similarity between solutions such that they are no longer considered unique. We define this in terms of decoding bit-error, where observed decoding bit-error values below .4 will be considered collisions.

By using model weights as keys, we are incorporating an extra layer of security on top of just the use of steganography. If we put all the layers of security for our steganographic system together we have our steganographic algorithm, its unique weights, and encrypted message (before steganographic encoding). In order for an attacker to read our original secret message, first they would have to recognize that steganography was used, which breaks our required security assumption for the ML-based steganographic model. However, even though the model is technically broken at this point, the attacker cannot yet read the original message and there is hope of protecting the long-term security of the message. They would need to identify the particular algorithm used and if it was a ML-generated algorithm, they would also need the specific weights for all neural networks. Finally, if they were able to obtain that information, they would need to utilize some form of cryptanalysis to obtain the encrypted message. By using a combination of unique steganography and provably secure cryptography, we increase the cost to the attacker.

3.2.5 Usability

Usability is the collective measurement of several other elements including capacity, reliability, bandwidth, processing power, and user interface. In terms of capacity, the model must be able to embed a message of a "reasonable" length. Given the maximum size of a twitter "tweet" is 280 characters, we will require a required capacity of 280 characters as a heuristic for user expectations of capacity. Average Short Message Service (SMS) lengths

vary between languages and age groups. If Unicode Transformation Format - 8 Bit (UTF-8) encoding is used, UNICODE encoded characters take up from two to four bytes each, which still allows a minimum of 70 Chinese characters, likely more, which is much longer than the average Chinese sentence, far exceeding our requirements [37]. Research from 2014 on machine translation has shown that the average sentence length in Chinese microblogs is about 25 characters [37]. Therefore, our model could theoretically embed at least two and up to five average microblog posts, which is more than enough space to convey a concise message.

As a sub-component of usability, a certain degree of reliability as a feature is assumed in most modern technical applications. However, reliability in our case depends upon the accuracy of the decoder to decode the embedded message in a stego-image. Because we are considering ML-generated models, loss metrics will improve gradually as training is performed. In many cases, obtaining a loss of zero is impossible, which means that there will be some small level of loss or inaccuracy with models generated using ML. Therefore, we require a maximum limit of .05 bit-error in message decoding. In other words, our model must be able to decode a given embedded message from a cover image with at least 95 percent accuracy. Researchers have shown that using error correcting codes with ML generated steganographic models can reduce small decoding bit-errors to zero [25]. Therefore, we assume that error-correcting codes are used in tandem with the maximum .05 bit-error requirement.

Bandwidth and processing power are also both important considerations. In order to generally bound the problem, we require that bandwidth and processing power requirements be commensurate with other popular, currently deployed mobile applications. This level of bandwidth is sufficient because our model should not require any extra bandwidth, as encoded messages are in the form of modified pixel values and do not increase the amount of data transmitted. The only potential challenge to constraining processing power requirements to currently deployed mobile applications is the introduction of ML. However, as stated above, as long as increases in power consumption do not significantly impact the user's experience, at least to the extent that they do not deter the user from using the application, the processing power requirements are somewhat flexible.

Latency is another key sub-component of usability. Latency requirements for our model

are similar to popular messaging applications. The main requirement is that messages are delivered and accessible in near-real time, on the order of seconds. We have a higher tolerance for initial key exchange latency but in order for this model to be usable, it should not significantly detract from the user experience (considerations for initial distribution are discussed more in Section 5.2.6). By increasing the latency tolerance, we enable the ML generation of a sufficiently unique steganographic solution, thereby increasing the difficulty of steganalysis. In other words, ML takes time but can produce a unique solution, which protects users from would-be attackers.

In regard to user interface and experience, it is unreasonable to expect to attract a high number of users with a complex and difficult-to-use program. Therefore, this model utilizes existing platforms to simplify the problem of initial adoption. However, users will still be required to download a plugin-type application that uses a simple GUI. Many messaging applications, especially Chinese-developed applications, provide limited developer support, especially for developers interested in integrating features that may be at odds with the original developer's or the local government's goals and priorities. Therefore, our model will be flexible enough to be implemented with different platforms and this research will focus more on the model itself than implementation details.

3.3 Model Selection

In order to meet the above requirements, we selected a pre-existing machine learning steganographic model as a template. We selected the "HiDDeN" model produced by researchers Jiren Zhu, Russell Kaplan, Justin Johnson and Li Fei-Fei of Stanford University [25]. In a 2018 paper entitled, "HiDDeN: Hiding Data With Deep Networks," the researchers showed how neural nets can be used to train both an encoder that is capable of producing stego-images free from visually perceptible perturbations and a decoder that can decode the hidden message from the stego-image with a high degree of accuracy [25]. There were several implementations of the researchers' code available on GitHub, but, ultimately, we chose to use a PyTorch implementation created by "ando-khachatryan" due to Python usability and the code being easily modifiable [38]. The included code allowed for a wide range of experimentation which included a long list of command line arguments, some of the most important being message length, number of training epochs, and batch size. Although this model fits several of our requirements "out-of-box," it is computationally demanding

and training our models until convergence initially took nearly a week on high-end last generation NVIDIA GPUs.

Using the "HiDDeN" model as a template, we make multiple, significant modifications to the original code which allow us to perform a number of experiments which aim to evaluate each of our requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Model Evaluation

In order to evaluate the effectiveness of our chosen steganographic algorithm for implementation within a broader censorship subverting framework, we performed various experiments listed in Table 4.1. Much of the experimentation involved modifying the implementation of the HiDDeN algorithm to achieve our model's goals outlined in the previous chapter. More specifically, we sought to answer the following questions:

- Can a ML-generated steganographic algorithm be ratcheted in a way similar to cryptographic ratcheting, and if so, what is the best mechanism?
  - If so, this would enable a steganographic form of forward secrecy.
- Can the HiDDeN algorithm be used to satisfy our capacity, message length, and visual imperceptibility requirements?
  - If so, the HiDDeN model has potential to be incorporated into an anti-censorship steganographic system.
- Do any modifications to the original implementation affect resistance to steganalysis?
  - If so, this would limit the algorithm's appeal and require further modification and testing.

4.1 Feasibility of Training to Generate New Models

The following section describes experiments conducted to evaluate the model's initial suitability to meet the unique solution requirement specified in Section 3.2.4. Because the terms "intra-model" and "inter-model" are used frequently throughout the chapter, they are defined here. When we refer to the intra-model bit-error, we are referring to the observed bit-error when using the encoder and decoder from the same epoch of the same model. If the ML model is adequately trained and converges to a solution, this number should be close to zero, indicating that the model can generate and decode its own messages with high accuracy. When we refer to the inter-model bit-error, we are referring to the bit-error observed when decoding stego-images with a decoder taken from one model, and an encoder taken from another. If the models are truly "separate," meaning that they utilized different

Experiment Description	Goal	Results	Significance
1. Feasibility of Additional Training to Generate New Models			
1.1. Inter-epoch decodability	Determine intra-model (single model) encoding/decoding compatibility between training epochs	Low bit-error = high compatibility	Extra training is not a good approach for ratcheting model
1.2. Inter-model decodability	Determine encoding/decoding compatibility between epochs between two separately trained models	High bit-error = low compatibility	Separately trained models converge to unique solutions
2. Message Length and Capacity			
2.1. Maximum message length testing	Determine maximum feasible message length	Any length past 64 bits takes too long to train (multiple days +)	Other techniques to increase encoding capacity are needed.
2.2. Image split into many small tiles and tiles encoded individually	Determine feasibility of tiling	Visually perceptible but very subtly	Tiling approach could be effective solution for increasing model encoding capacity
2.3. Calculate differences between cover and stego-image (sub-experiment of tiling feasibility)	Determine how model is modifying pixel values and color channels	Greatest changes to color channels appear where that color is least present, changes are nearly binary in nature	Edge smoothing techniques may be effective in decreasing visual perceptibility without corrupting encoded message.
3. New Model Generation by Modifying Model Weights			
3.1. Add random noise to encoder/decoder final layers to create new model	Determine effectiveness of adding random noise to select neural net layers to ratchet model.	Somewhat effective, approach breaks down after multiple ratchets, introduces too much loss. Star-generation model approach is fairly effective	Limits of this approach are worth investigating.
3.2. Determine noise thresholds for creating new model	Determine if noise-adding to select layers approach is worth continued experimentation.	Could not find suitable threshold, bit-error too high and training time too long	Not feasible for ratcheting: slow convergence, too many collisions.
3.3. Reinitialize layers to create new model	Determine effectiveness of re-initializing neural net layers to ratchet model	Very effective (fast convergence, low loss and bit-error)	Potentially feasible for ratcheting. Key exchange / bootstrapping is still a problem (how to send weights securely over public channel). Production of weights is restricted to one party.
4. Steganalysis Vulnerability			
4.1. Encode/decode sample image 1000 times over three separate models, determine distribution of errors	Determine distribution of errors over 30 bits	Errors are not uniformly distributed	Variation of this technique could be used in steganalysis (detection)
4.2. Encode constant, known plaintext in 2000 sample images with model A, decode with model B, determine distribution of errors	Determine distribution of errors over 30 bits	Errors are not uniformly distributed	This technique could be used in steganalysis
4.3. Generate 2000 sample cover images, decode with model B against known plaintext, determine distribution of errors	Determine distribution of errors over 30 bits	Errors are distributed much differently than for decoding stego-images	Combined with experiment 4.2 technique to create feasible attack, results in huge vulnerability for HiDDeN and other ML steganography
4.4. Perform statistical steganalysis attacks against HiDDeN, SGM, and ratcheted models	Determine effectiveness of attacks	Attacks not particularly good at detecting steganography	Modified models still robust against common, statistical steganalysis techniques

Figure 4.1. Table of experiments and results

initialization seeds or they have been effectively ratcheted, the observed bit-error should be very close to .5, mirroring the bit-error one would achieve with random guessing.

4.1.1 *Inter-epoch Decodability*

Following initial setup, our first experiment was designed to determine if it was possible to take a pre-trained model and ratchet it forward in a way that generates a new, unique solution that is not compatible with other models, including the pre-trained model it was generated from. In order to determine if it is feasible to modify model weights to create new, unique encoders and decoders that have no backward compatibility with previously trained encoders and decoders, we first analyzed the ability of the HiDDeN model to encode and decode stego-images produced by encoders and decoded by decoders taken from different

training epochs for a single HiDDeN model. In other words, we assess if extra training causes a model to converge to a new steganographic solution. Our test involved producing a stego-image by encoding the test cover image with a randomly generated 30-bit string, decoding the resulting stego-image with our selected decoder and measuring the bit-error by rounding the decoded values and then comparing them to the original randomly generated string [38]. For example, a cover image was encoded with the encoder from epoch 1, producing a stego-image, then that stego-image was decoded with decoders taken from every epoch tested, 1 through 300. We used the PyTorch implementation of HiDDeN's authors created by user "ando-khachatryan" on GitHub as a framework for our tests. Most of our experiments are heavily modified versions of the "test_model.py" program [38]. This program generates a random bit string, encodes it into a user-specified cover image, decodes the message, and determines the bit-error.

For each encoder and decoder pair, this test was performed ten times and the final average bit-error was calculated across all runs. A perfectly decoded message with no errors has a bit-error of 0, the minimum, and at the other end of accuracy is a bit-error of .5, the maximum, which is equivalent to random guessing. A bit-error of .5 is the maximum because if one were to achieve a consistent bit-error of 1, where every bit is decoded as the exact opposite of the corresponding bit in the original message, we could easily flip all the bits in the decoded message to achieve a bit-error of 0. In other words, a bit-error of .5 signifies high entropy and no discernible connection between input and output values. Because a bit-error of .5 is synonymous with maximum entropy, for all figures describing experiments in this paper, the maximum possible bit-error is .5. Specifically, for the reasons described above, any actual bit-error values exceeding .5 were converted to the absolute value of the difference between that value and .5.

The goal of this experiment was to determine if extra training could be used to ratchet a model's weights forward to create a new steganographic solution, preventing the decoding of images generated by an "older" model when using the decoder from a "newer" model, a subsequently trained epoch, and vice versa. The results were unsurprising in that they confirmed that there is a high degree of encoding and decoding accuracy across training epochs belonging to a singly trained model. This is unsurprising because during training, each subsequent training epoch incrementally reduces loss and bit-error and thus converges to the same local minimum or solution. In other words, additional training, especially for a

model that has converged or nearly converged to a usable steganographic solution, is not a viable solution to ratchet our model forward to generate a new, unique model.

Figure 4.2 shows encoder/decoder combination bit-error declining with additional model training. Higher number epochs are those that have undergone more training and thus are better able to decode stego-images produced in that epoch and in previous epochs. With the exception of some of the earliest epochs, 0 through 30 or so, later epochs can decode stego-images produced by encoders from any epoch belonging to the model with a great degree of accuracy (very low bit-error). In other words, the asymptotic, downward sloping line shows that most encoder/decoder pairs are compatible. After about 50 epochs of training, inter-epoch, which is equivalent to intra-model, accuracy increases to over 95 percent. These results therefore rule out extra training on a single model as the sole means to ratchet our steganography algorithm forward. In this case, even an extra 200 epochs of training did not produce a mutually unintelligible encoder/decoder pair, because as was mentioned above, all the epochs are converging toward to the same solution.

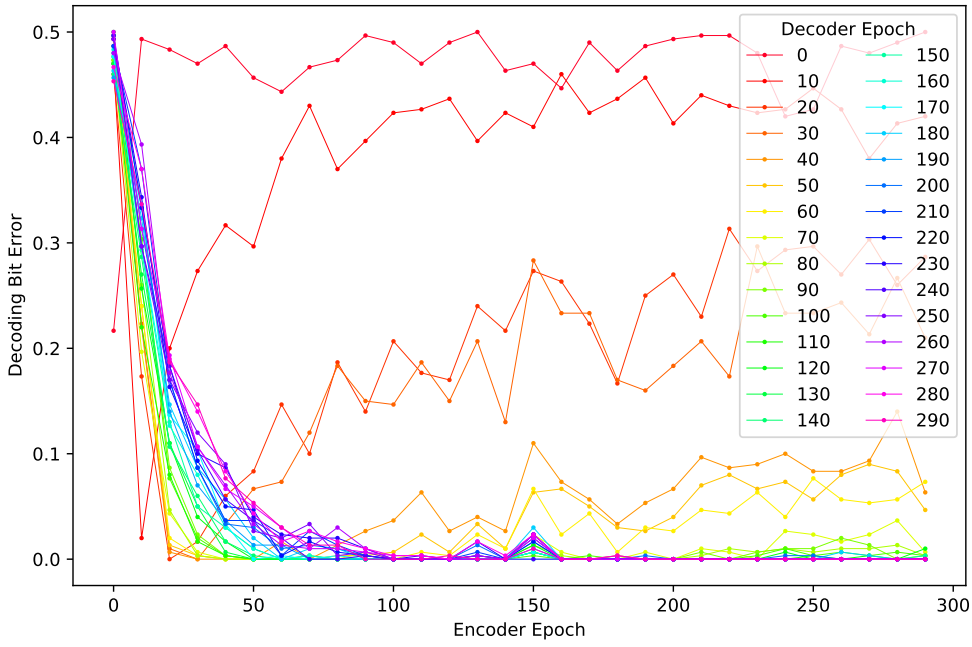


Figure 4.2. Low intra-model bit-error for all encoder and decoder combinations

Figure 4.3 contains the same data as Figure 4.2 but displayed in a different way. It can be seen that for all encoder and decoder combinations above 100 epochs, the graph is solid blue, indicating the corresponding bit-error for each of those combinations is less than .1. On the other hand, the areas of the graph with cyan and red indicate high bit-error and low compatibility, all involving encoder/decoder combinations with either the encoder or decoder being low in number. To summarize, both Figures 4.2 and 4.3 demonstrate that as the model is trained, the encoder/decoder combinations with higher number epochs generally show greater compatibility with one another.

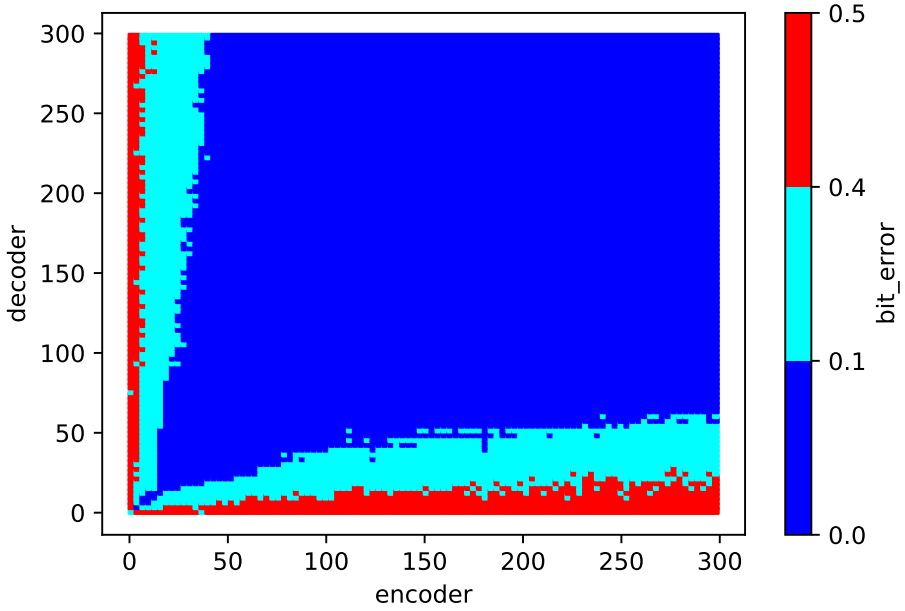


Figure 4.3. Threshold values of .4 and .1: blue indicates usable level of compatibility, cyan indicates a collision (some compatibility), and red indicates virtually no compatibility

4.1.2 *Inter-model Decodability*

The goal of this experiment was to determine the ability of separately initialized and trained models to encode and decode stego-images produced by one another: in simpler terms, we sought to answer the question, “are HiDDeN models unique”? Although, the original HiDDeN researchers did not describe this exact experiment, they did perform experiments

which showed “image distortion patterns produced by each trained model are unique” [25]. Rather than evaluate the ability of separately generated and trained model’s ability to decode each other’s stego-images, the researchers generated several models to train a state-of-the-art machine-learning enabled steganalyzer [25]. It turns out that the steganalyzer trained on a single model’s steganography was unable to detect the use of steganography by a separately trained model, even though both models were generated using the HiDDeN algorithm, with the only difference being the initialization seed [25].

For this second experiment, we trained two models with message length of 30 bits for 300 epochs and then evaluated the ability to encode and decode a randomly generated 30 bit message using an encoder from one training epoch of one model and the decoder of a training epoch from another other model. The experiment was conducted in a very similar way to the previous experiment in that we tested every combination of encoder and decoder pairs. The main difference between the two experiments was that in this case, encoders and decoders were loaded from checkpoints of separate models rather than subsequent training epochs belonging to the same model. Results displayed in Figures 4.4 and 4.5 show that inter-model decoding bit-error is consistently high, with a minimal bit-error for this experiment being .39. The darker colored squares in 4.5 indicate a larger bit-error. Because all encoders were matched with decoders from different models, it is unsurprising that high bit-errors were observed for every tested combination. The graph was generated using a diverging color-map to emphasize the near-randomness of the observed average bit-error for each encoder/decoder combination. The fact that there are very few light colored squares indicates that the decoder from even the most compatible encoder/decoder combinations has a very limited ability to decode stego-images produced by another model. Although this experiment was performed at a small scale, results showed each model unique it its ability to encode and decode only images produced by the model itself. Our experiment’s results in Figures 4.4 and 4.5 support the hypothesis from [25] that each trained model is unique by demonstrating a near complete inability to perform inter-model encoding and decoding of images. By demonstrating the inability of one model to decode stego-images produced by another model, we showed the potential for developing a ratcheting mechanism which generates unique steganographic solutions.

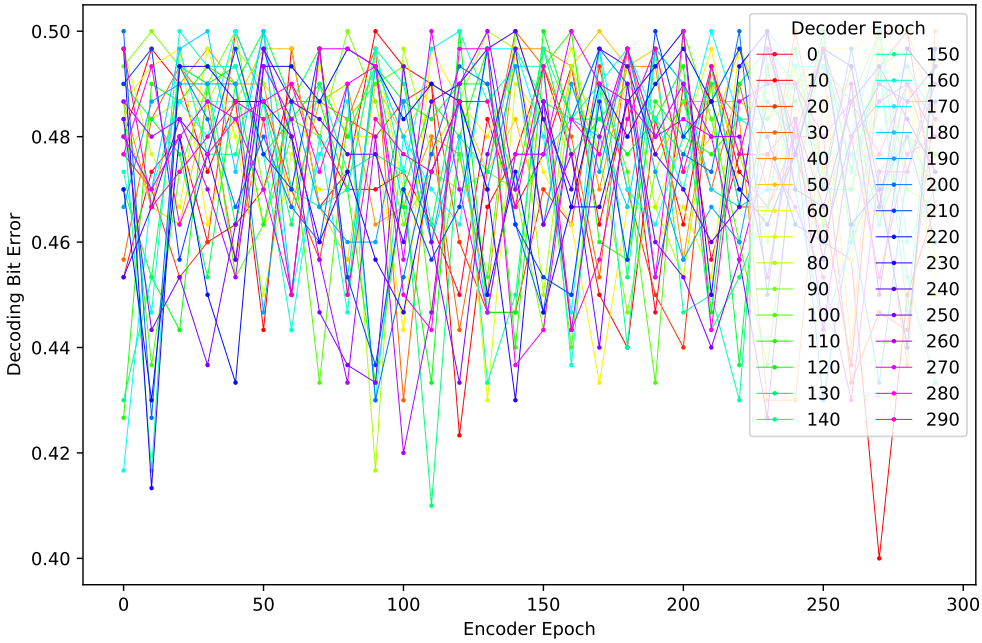


Figure 4.4. High bit-error for encoders and decoders from different epochs across multiple models

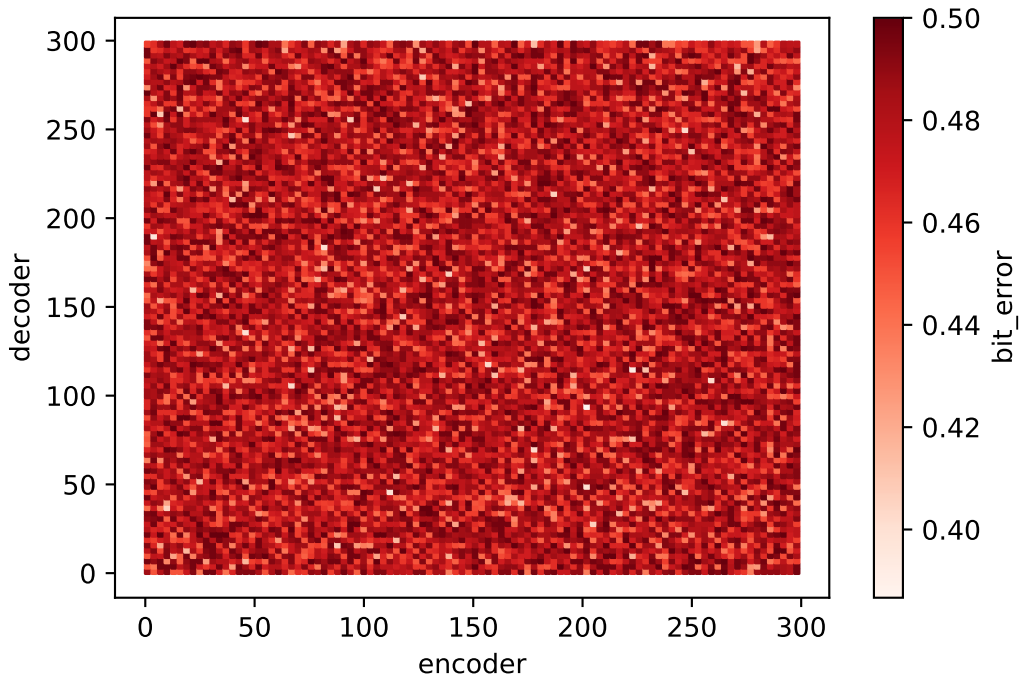


Figure 4.5. Minimum bit-error greater than .4. No inter-model decoding compatibility.

Our results showed that models are not only uniquely detectable, as shown by the original researchers who developed the HiDDeN algorithm, but that they are also uniquely decodable. In this context, a model can be considered uniquely detectable if knowing the underlying algorithm used does not affect an attacker's ability to identify the use of that particular model. Researchers showed that ML steganalysis techniques are only effective per individually trained model and not across all HiDDeN models. In order words, if there are two HiDDeN models, equivalent in all ways except for initialization seed, then effective steganalysis techniques on one of the models will not inform effective steganalysis techniques for the other.

Unique decodability refers to stego-images produced by each individually trained model only being decodable by that specific model. In other words, stego-images produced by one model cannot be successfully decoded by another. In a simple non-multicast communications

network, we need to create a model with unique decodability for each pair of users. In our model, the weights of the final layers of the encoder and decoder serve as a steganographic key. Therefore, we looked to altering model weights as a potential means to ratchet our model forward. However, our ability to do this depends heavily on the "key space" for the weights in the last layers of our encoder and decoder nets. Key space in this context refers to the number of unique weight selections such that valid models that do not collide with one another.

Before estimating the key space, we chose to test if ratcheting our model forward just one time was even possible. It is important to note that because of the computation power and time required to generate a single usable model, training each model from scratch for use between two users is not feasible. For example, the time it took us to train a "usable" model was several hours on relatively new GPUs (NVIDIA TITAN X). For our specific model we achieved a bit-error less than .05 and a message length of 30 bits on 128 by 128 pixel cover images. For training data, we utilized the entire 2017 COCO data-set as our training set. Of course, there are ways to further reduce training time, such as reducing the size of our training set, however, we mention the amount of time to emphasize the general degree of time and energy required to train a new model from scratch. As we mentioned earlier in the paper, 30 bits of message length is far from usable for our purposes; we need at least a few thousand bits to fit a few sentences of ASCII-encoded text.

Therefore, even using state-of-the-art hardware, we cannot realistically ratchet our model as it requires training from scratch every time. Developing a new model from scratch to generate model weights to be used in a kind of key generation and exchange mechanism would hurt overall stego-system usability because of time, power and computational requirements. Additionally, training a functional model on a smartphone, or on multiple smartphones in a distributed setup, is likely not possible in a reasonable amount of time. For example, the Signal mobile app allows users to ratchet encryption keys almost instantaneously. Although this is not a requirement for our model, users cannot be reasonably expected to wait hours or days while background applications are training a model because energy consumption is a limiting factor for mobile devices. Burning through a large percentage of a smartphone battery's capacity as users establish a new communications path is extremely inefficient and not feasible for mass adoption. However, as previously shown, it appears that each separately trained model is uniquely decodable. Ideally, we could distribute a pre-trained base model

that can be minimally modified to be uniquely decodable.

4.2 Message Length and Capacity

The following section describes experiments which evaluate our model's ability to meet the character requirement referenced in Section 3.2.5 without affecting the visual imperceptibility requirement outlined in Section 3.2.1. Additionally, we explore the specific techniques our model can learn to hide data and a method to increase encoding capacity.

4.2.1 *Maximum Feasible Message Length*

In order to be feasible, our model should scale to longer message lengths in order to decode at least 280 ASCII characters as outlined in the previous chapter. The original authors of the model used short message lengths (usually 50 bits or less) to perform most of their experiments, perhaps in an effort to minimize training time. However, our model must be able to handle a total message length of 2240 bits, equivalent to 280 bytes, with one byte per character, as described in Section 3.2.5. We attempted to scale the model from a default 30 bit message length to a 280-byte message length with 128 by 128 images, which still falls well below the researchers claimed capacity of .2 Bits Per Pixel (BPP) with a BPP of less than .14. However, initial training proved extremely slow with the bit-error only dropping to .45 after nearly a month of continuous training on NVIDIA TITAN X and GTX 2080 graphics cards. Although the model may eventually converge to a solution, given the exorbitant amount of time required to train such a model, our efforts became focused on other solutions to achieve a usable message length. After much trial and error, we settled on using a message length of 64 bits which we determined to be a good balance of training time and capacity. Although 64 bits is far from our stated capacity goal of 2240 bits, the advantage of a 64 bit model lies in the relatively small amount of training time. The extra training time required to train a 64 bit model over a 30 bit model is several hours whereas the a 128 bit model requires at least a day of extra training time. Longer message length models, in addition to failing to come close to the stated goal of 2240 bits, also require a significant amount of training, considerably hurting model usability in two different ways. If we can find an approach or specific implementation in which we can use frames or crops/tiles that are embedded with 64 bits of message data each, we could satisfy our model's requirements

with as few as 35 frames or tiles. A 30 bit model would require at least 75 such frames or tiles.

4.2.2 *Effectiveness of Tiling Approach*

Although 64 bits is a big improvement on 30 bits, it still only allows for encoding 8 ASCII characters, meaning our program would not be able to encode the timeless phrase "Hello World" and be limited to just "Hello Wo." We identified two possible solutions to increase overall image capacity. The first was to utilize animated GIFs or short videos and encode each frame with 64 bits of data. With only 35 frames, which could be around one second or less[2], we would be able to meet our minimum capacity requirement of 280 characters. However, this approach would involve dealing with a different image format, potentially introducing new complications. For example, we would need to use existing GIFs or short videos as our cover media, which significantly reduces the flexibility and usability of our model. Additionally, there is the possibility that the steganography would be more visually perceptible in this format. For example, if edge effects do exist, it may become more obvious when the frames are displayed one at a time over one another. Therefore we decided to pursue a second approach which involved slicing a cover image into many smaller tiles and encoding each tile with a proportionate chunk of the overall message. In our experiment, we first cropped an image into a square, sliced the cropped image into tiles, and then encoded each tile with 64 bits of data, and finally stitched the image back together. In practice, on the receiving end, the image would be sliced again, each individual tile decoded and the encoded message from each tile would be concatenated together to reproduce the original message.

Our first experiment was designed to determine if this approach was visually perceptible. We trained multiple models with a 30 bit message length and crop sizes ranging from 16 by 16 to 128 by 128. We then took a cover image, 256 by 256 or larger, sliced it into smaller tiles and encoded each "tile" with 30 bits worth of information. In the case of using a 256 by 256 cover image with 16 by 16 tiles, our total capacity is 256*256/(16*16) = 256 tiles holding 30 bits each, 256*30 = 7680 bits or 960 ASCII characters, far exceeding requirements for our overall model. We used sample images similar to those in the COCO data set in terms

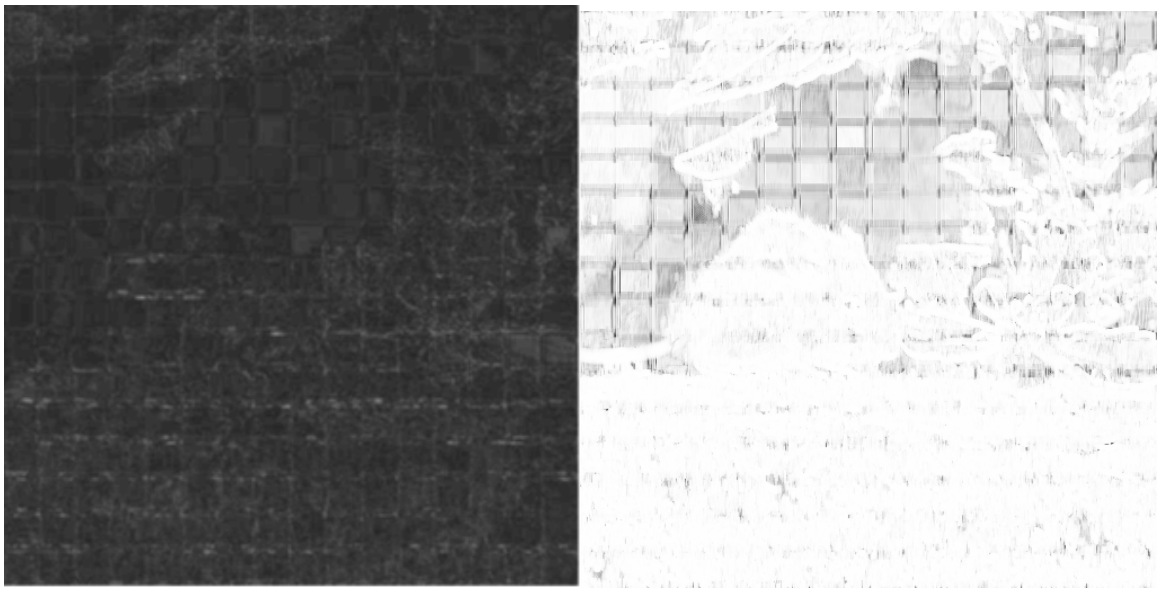
[2]Per the GIF98a specification, by setting the delay time in a GIF98A image, one can create GIF images with a wide range of frame rates, from over 60 frames per second to less than one [39].

of general characteristics like subject, lighting, size, etc. We chose to use images that were typical representations of human life and that are the kinds of images commonly shared by users of messaging applications, rather than solid colors and/or geometric patterns. We tested different-sized images ranging from 128 by 128 to 1024 by 1024. In each case, we kept the capacity ratio well below the HiDDeN authors' standard of .2 BPP.

Results showed that generally, the use of smaller tiles, such as 16 by 16 and 32 by 32, was more visually perceptible than larger tiles such as 128 by 128, but the differences in visual perceptibility were not dramatic for tile sizes above 16 by 16. Additionally, increasing the message length from 30 bits to 64 bits did not have a noticeable impact on visual perceptibility. Most stego-images produced with the tiling approach showed faint box-like artifacts that were noticeable in areas of the image with low entropy. For example, in our test image of a shrew under a leaf, there was a dark, mostly mono-color area of the image. Such areas that appear "smooth" and have very little texture seemed the most prone to artifacts, but further testing is necessary to make any stronger claims about the relationship between the types of images and textures used for cover images and artifacts observed in the resulting stego-images. Although results did produce some visual artifacts, the initial results were positive enough for us to continue testing the viability of this approach. Figure 4.6 shows the limited visual perceptibility of the steganographic tiling approach, even with a small, 16 by 16 tile size. Although this test was repeated for two other images and showed similar results, this experiment is merely an example meant to demonstrate potential feasibility for further testing.

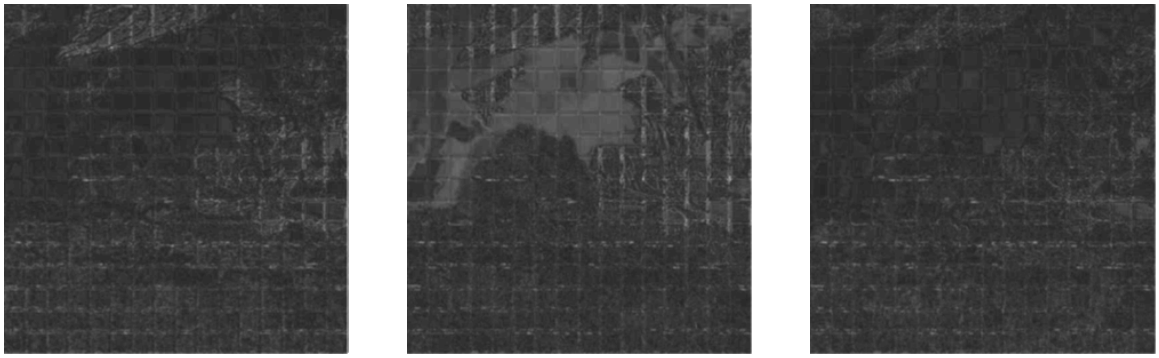
(a) Cover image (b) Stego-image

Figure 4.6. Side-by-side comparison of cover image and stego-image using tiling approach



(a) Pixel differences (RGB) (b) SSIM

Figure 4.7. Side-by-side comparison of pixel differences and SSIM between cover image and stego-image. (a) has been edited to increase brightness and contrast for viewability/aesthetic purposes.



(a) Red (b) Green (c) Blue

Figure 4.8. Side-by-side comparison of differences in each color channel between cover image and stego-image. Lighter colored areas indicate greater changes in pixel values. (a), (b), and (c) have been edited to increase brightness and contrast for viewability/aesthetic purposes.

4.2.3 *Determine where and how model encodes data in images*

Inspired by mostly positive results in our testing of the tiling approach, our next experiments were designed to identify issues with and solutions for improving the tiling approach by analyzing exactly how the stego-image is modified. In other words, we wanted to provide at least a limited view of the black box that is our steganographic ML-algorithm. Three sample images that differed in content, color, and texture were used to test the hypothesis that low entropy areas of an image are most prone to producing visually apparent "edge-effects" in the stego-image. We computed the difference in pixel values for all color channels in our cover images and stego-images. We also computed the SSIM between cover and stego images [40]. Figures 4.6, 4.7 and 4.8 show the parts of the image affected by the steganography. Unsurprisingly, due to the minimal visually apparent differences between the cover and the stego-image, the changes made to the cover image by the steganographic model were small. Average pixel value changes per color channel were 4.02, 6.46, and 4.40 for blue, green and red channels, respectively. In the case our original shrew image, the background has high levels of green and as our color channel analysis shows, the changes in pixel values in the green channel were greater than changes in the other color channels. This suggests that the HiDDeN algorithm makes the greatest changes to the most represented color channels in the original image. However, because the algorithm behaves this way, large changes are made to areas of the image with low entropy. As seen in Figure 4.8, the highest concentration of changes in pixel values was in the area of the image containing the green leaves. Because these changes were not evenly distributed throughout each tile, the resulting stego-image contains a faint, but visually perceptible, cross-hatch pattern.

However, because each model generated by HiDDeN appears to be unique, it is unclear how susceptible this model may be to statistical steganalysis of color channel differences. The tiling process likely makes steganalysis easier due to the fact that the tiling frequency itself is likely measurable via spectral analysis and can be used to differentiate between stego and non-stego images (for more details see Section 5.1.1). Because the models generated by HiDDeN appear to be unique, a claim made by the original authors and supported by both their experiments and ours, described in Section 4.4, the specific ways in which cover images are modified or distorted should differ across models. However, exactly how much these models differ in the specific ways they modify pixel values is unclear without further testing. Therefore, further testing is necessary to determine the commonalities that exist between

image distortion techniques utilized by different HiDDeN models. Determining potential similarities between unique solutions could provide insight into model vulnerabilities against steganalysis.

4.3 Modifying Model Weights to Generate New Models

In this section we explore several techniques to modify model weights to generate a unique solution, as first described in Section 3.2.4.

4.3.1 *Effectiveness of Ratcheting by Adding Random Noise to Encoder/Decoder Weights*

Our next experiment involved fine-tuning model weights from both the encoder and the decoder by freezing all the neural networks' layers except the last and then adding noise to the weights of the last layers and performing additional training. The goal was to modify the weights of the last layers in such a way that we could converge to a new solution. If we could converge to a new solution, we would have a model that cannot decode stego-images produced by other models with anything greater than 50 percent accuracy, and vice versa, from previous iterations of the same base model. In this way, the weights in the last layers of the encoder and decoder form the steganographic "private key."

Our first method of adding weights to the last layers of the encoder and decoder was to first calculate the mean and standard distribution of those layers' weights and to generate normally-distributed random noise based on those statistics. Once we added noise and trained this new model for ten iterations, we then tested it for compatibility with previous epochs of the same model from before noise was added to the weights. Immediately after noise was added, several error metrics increased drastically such as the encoder MSE, loss and bit-error between pre-noise added and post-noise added epochs. This indicated to us that adding weights to the model significantly impacted the model's functionality, particularly the basic ability to encode and decode a given message from a given cover image. However, as the model trained, the loss metrics for post-noise added epochs slowly decreased. After scaling the weights down by half by performing element-wise multiplication of each element in the noise array by .5, we discovered that the new model was usable, in that it achieved a bit-error very close to zero, .01, after only 1 epoch of additional training, and that all

trained epochs after noise was added to the last layers of both the encoder and decoder were incompatible, seeing a decoding bit-error of .48, with epochs from before noise was added. Strictly speaking, an image encoded with a encoder taken from a pre-noise-added epoch cannot be decoded with a decoder taken from a post-noise added epoch: adding random noise to the final layers of the encoder and decoder essentially produces a completely new model with its own unique steganographic technique.

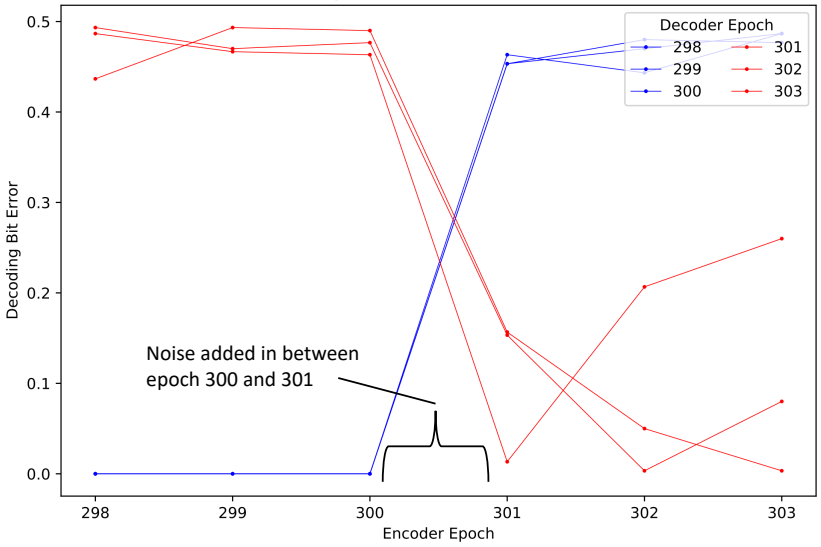


Figure 4.9. Effect of adding noise to last layers of encoder and decoder neural net and additional training on decoding bit-error

Figure 4.9 shows that adding noise and performing one additional epoch of training produces a completely new model. This experiment demonstrated the ability to add random noise and perform additional training to ratchet our model forward. In order to confirm that we can reliably ratchet models forward by adding random noise to the last layer in the encoder and decoder neural nets, we needed to be able to reproduce the results. Specifically, we needed to show that by adding noise, we can create a large number of new models that has no backward compatibility with epochs from before the noise was added nor with any other models.

4.3.2 *Determine Minimum and Maximum Thresholds for Adding Noise to Model Weights to Obtain New Solution*

The following experiments were designed to test the appropriate level of noise to simultaneously maximize convergence speed, maximize inter-model bit-error, and minimize intra-model bit-error. For example, if too much noise is added, the model either will not converge or be slow to converge. On the other hand, if too little noise is added, the inter-model bit-error will be too low as each model's solution is too similar to other existing models. In this situation, we may observe speedy convergence but also many collisions, in the form of low inter-model bit-error.

Although a scaled-down version of random noise, with a scaling factor of .5, was effective in the previous experiment, further testing showed a noise scale of .5 and even .75 to produce a model that has a significantly lower average inter-model bit-error than those shown in Figures 4.4 and 4.5. To be specific, the scaling of weights was conducted by multiplying the standard deviation of the encoder's and decoder's final weights by .5, respectively. Rather than a mean bit-error of .5, which is what is required to have a sufficiently different model, the actual average bit-error was closer to .4. Only after increasing the noise scaling to 1 did we see results showing an average bit-error approaching .5.

Additionally, it is important to note the importance of the optimizer in minimizing training time and converging to a new solution. The goal of this experiment and several following experiments is to produce a new, unique model from a previously trained model as quickly as possible. In the first several iterations of these experiments we did not adequately predict the importance of the optimizer and the optimizer state. A ML optimizer is used to identify and adjust the parameters that have the greatest role in reducing a given cost function. In ML, the cost function calculates the difference between predicted and observed values and serves to minimize some metric, such as loss, error, or cost. Certain optimization algorithms are better-suited towards certain applications and can significantly affect convergence speed.

The particular model we chose to modify, the original HiDDeN model, utilizes an Adam optimizer [25]. Adam is known for its typical "out-of-the-box" usability and minimal hyper-parameter tuning requirements [41]. One key characteristic of Adam is that it has adaptive learning rates for each parameter that are calculated by using decaying averages of previous gradients [41]. In other words, previously performed training informs how the algorithm

will take into account current training metrics. By adding random noise to the final layers of our encoder and decoder we are seeking to force the model to converge to a new solution, which is typically the opposite problem that most ML applications are designed to solve. Typically, ML models are designed to converge to the best possible solution, whereas in our case, we are attempting to converge to a completely new, unrelated solution. In other words, we are trying to "jump" out of a local minimum, which can be graphed as a sort of ravine, with the magnitude of the jump being the amount of noise we add. By adding noise and then continuing training, we try to find a new minimum. Therefore, once we add noise, we no longer want to take our previous optimizer gradients into account: we need to reset the state of the optimizer in order to converge to a new solution as quickly as possible. In other words, by resetting the optimizer, we are wiping the optimizer's memory of the old solution so that it can converge to a new solution more quickly. A stateless optimizer, which has no such memory, such as stochastic gradient descent, would not converge faster or slower based on previous values because those previous values are ignored.

The previous experiments were replicated with the addition of re-initializing the optimizer whenever random noise was added to the final layers of our encoders and decoders. As predicted, the models for which the optimizer was re-initialized converged significantly faster, 2 epochs versus 5 epochs, than those for which the optimizer was not re-initialized. From this point forward, unless explicitly specified, it can be assumed by the reader that any further experiments involving model generation by altering model weights included re-initializing the Adam optimizer.

4.3.3 *Re-initializing Neural Network Layers vs. Adding Random Noise*

Now that we had found a potential mechanism for ratcheting our model, we sought to determine if there were any other better approaches. We evaluated the effectiveness of adding random noise to our encoder and decoder final neural net layers against completely re-initializing those same last layers. Results showed that the models for which the final layers of the encoder and decoder were re-initialized, converged much more quickly, training for 1 epoch as opposed to 5 or more epochs, to lower bit-error solutions than those generated by adding random noise. Furthermore, results also showed that models generated in this way were also less susceptible to collisions than random-noise-added generated models. By collisions we are referring to generated models that have at least some ability to encode

and decode stego-images produced by one another, indicated by average inter-model bit-error below .4. When graphed, these semi-compatible models can be easily identified with lighter coloring, showing low bit-error rates, less than .4, between encoder and decoders of different models. In summary, by re-initializing the optimizer and both final layers of the encoder/decoder neural nets and performing additional training, we decreased the intra-model bit-error rates, increased inter-model bit-error rates, and decreased training time for model convergence. Specifically, we reduced training time by 3 epochs, from 5 to 3, while also observing a greater average decoding bit-error between all encoders and decoders. The random-noise adding method, shown in Figure 4.12, resulted in an average bit-error of .45 while the re-initialization method, shown in Figure 4.13, resulted in an average bit-error of .49, which is apparent in the lower number of collisions seen in the re-initialization method.

4.3.4 *Approximate Risk of Collision of Star-generated Models (Estimate Maximum Number of Unique Ratcheted Solutions From Same Base Model)*

After initial success in ratcheting our model forward one time, we sought to estimate the number of unique solutions we can arrive at after ratcheting our model forward one time. To clearly delineate the difference between ratcheting, which involves serially creating new models with an inability to go backwards, and generating multiple new models from the same pre-trained base model via re-initializing the final layers of our encoder and decoder nets, we shall refer to the latter as a Star-Generated Model (SGM). The name comes from the star pattern created by generating many models from the same base model as shown in Figure 4.10. Figure 4.11 shows how our ratcheted models are generated and will be discussed in the next subsection. Additionally, it should be noted that the following test of the SGM was performed without resetting the optimizer as this experiment was performed before the optimizer re-initialization experiment. Results are shown in Figure 4.12. For comparison, Figure 4.13 shows results for the same experiment except with re-initialization of both the optimizer and parameters, highlighting the importance of both re-initialization procedures.

Like in our first ratcheting experiment, for this SGM experiment we start with a pre-trained model and add random noise to our final layers of the encoder and decoder nets and perform additional training. We started with the same base model from the previous

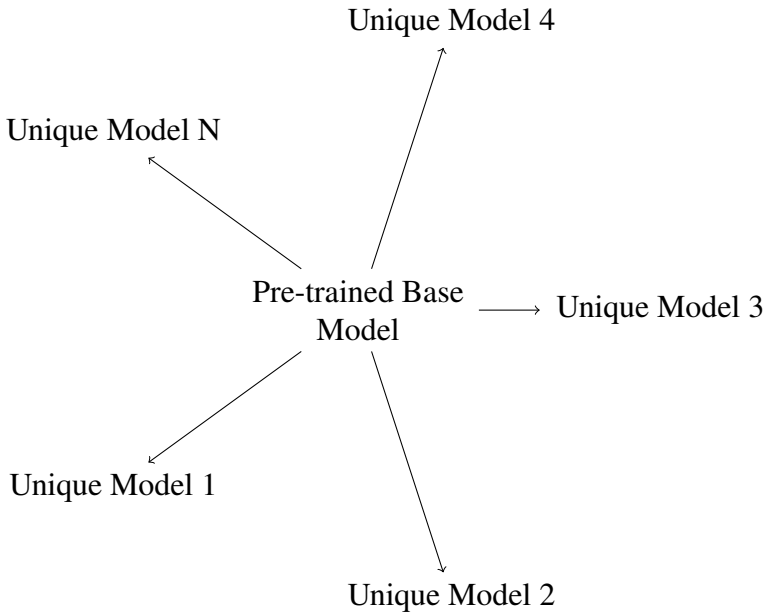


Figure 4.10. Star-generated model

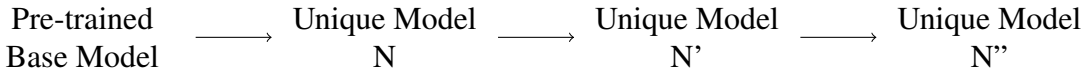
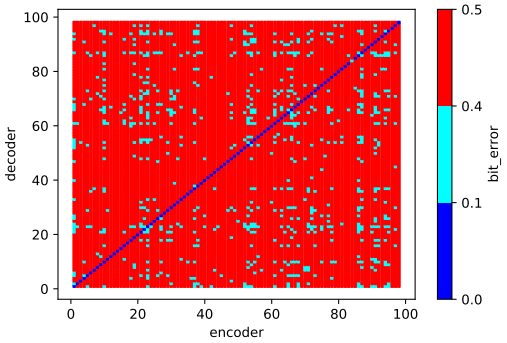


Figure 4.11. Ratcheted model

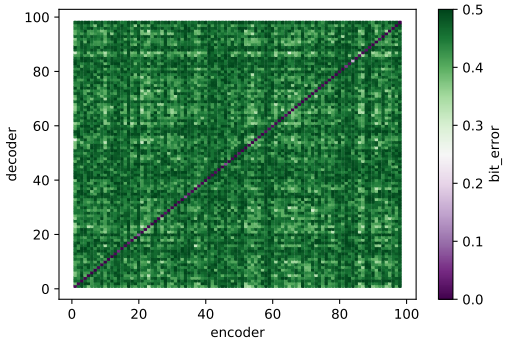
experiment, added random noise scaled by .5 and trained for 2 epochs 100 times. The goal of this experiment was to generate 100 new solutions by adding noise and training. We then evaluated each model's ability to encode/decode images generated by each other. Results show the average bit-error between supposedly "unique" models is less than .5, close to .4, indicating that there is still some capacity to encode and decode between these models, which does not meet our requirements. Additionally not all models achieved a low enough bit-error to be considered usable (we require < .05). Some solutions still produced a bit-error approaching .1. Furthermore, certain solutions were much more compatible with many other solutions, with a cluster of inter-model bit errors as low as .15. These clusters are shown as clusters cyan dots in Figure 4.12. This means that our experiment produced models that still have some ability to decode the steganography generated by other models, which is counter to our goals. All of these results point to several potential issues with the experiment. One possibility is that the scaling value for the random noise was too

high (.5), which would explain the low average inter-model bit-error. Another issue, the sometimes high bit-error seen in a single model's encoding and decoding is that the model was not trained long enough. Finally, another possible reason for these results was due to not resetting the optimizer, which can affect a model's ability to converge to a new solution. For our next sub-experiment, in order to increase the average inter-model bit-error and reduce the intra-model bit-error, we chose to increase the scale of added random noise to 1 and increase the number of epochs trained to 5 from the 2 epochs we used in the SGM experiment.

Following the results of our optimizer re-initialization experiments, we reproduced the above ratcheting experiment with the addition of re-initializing the optimizer and re-initializing the final layers of the encoder and decoder rather than adding random noise. As shown in Figures 4.12 and 4.13, results were very positive as they showed a marked decrease in the number of model collisions, which can be identified as cyan dots in both figures, and decreased intra-model bit-error, which can be seen in darker purple dots along y=x in the diverging colormap sub-figure in Figure 4.13. Smaller intra-model bit-error indicates greater decoding accuracy and fewer collisions indicate the inability of separate models to decode one another's stego-images, a requirement established in Section 3.2.4. Although the bit-error threshold graph pictured in Figure 4.13(a) still contains 8 total collisions, 8 is far fewer than the numerous collisions visible in the corresponding graph contained in Figure 4.12. The diverging colormap graphs featured in both Figures 4.12(b) and 4.13(b) display the same data as displayed in (a) of their corresponding Figure but in a slightly different way. The diverging colormap uses a gradual scale that gives the reader a more detailed understanding of the relative differences in observed bit-error.

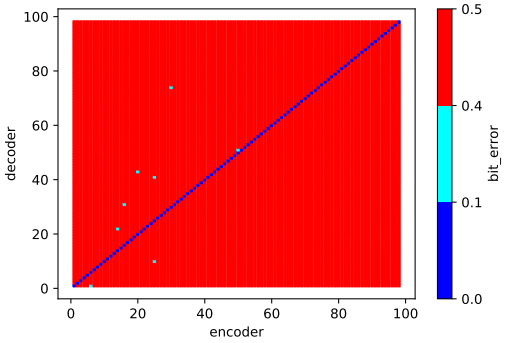


(a) Bit-error threshold

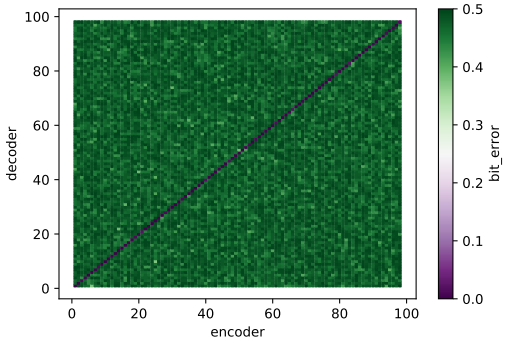


(b) Diverging colormap

Figure 4.12. Bit-error between encoders and decoders from star-generated models using random noise added to encoder/decoder final layers and without re-initializing the optimizer.



(a) Bit-error threshold



(b) Diverging colormap

Figure 4.13. Bit-error between encoders and decoders from star-generated models using reinitialized encoder/decoder final layers and reinitialized optimizer.

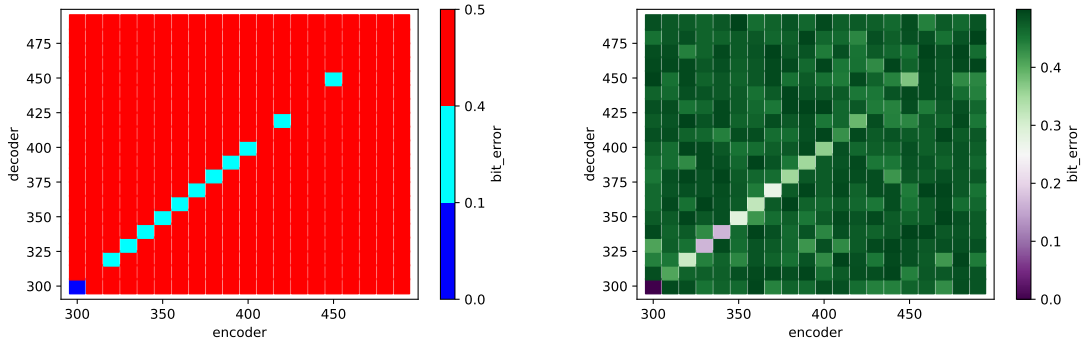
4.3.5 *Estimate maximum number of times model can be ratcheted forward*

In the last experiment we produced 100 models generated from the same base model and examined the number of valid models and collisions. As our model requirement was to not

produce collisions, we had limited success via our SGM method in that we only produced 8 collisions out 100 total models. Additionally, although we defined a collision as a bit-error below .4, a collision with a value of .39 is less concerning than one of .1, and the fact that we did not see any collisions with bit-error below .3 is notable.

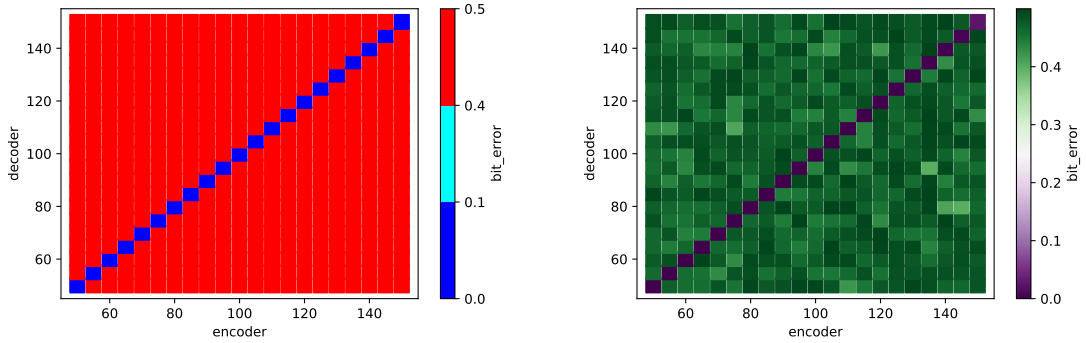
For the following experiments, like in the SGM experimentation, the initial versions of ratcheting experimentation did not involve re-initializing the optimizer. We tested the number of times a single model could be ratcheted forward by adding random noise to the final layers of a model's encoder and decoder. Specifically, we took the same pre-trained model as used in previous experiments and added noise with .5 scaling, trained for two epochs and then repeated until we ended up with a model that was not usable because the average bit-error between non-corresponding epochs was too low (.4), with edge cases where it dropped as low as .15. It is because of these results that we only trained for two epochs: the already low bit-error between non-corresponding encoder/decoder pairs was unlikely to increase with further training and it was apparent the noise scaling was too low. These results were similar to our initial test of the SGM using noise scaling of .5. We then increased the scaling of the random noise to 1 and trained for five epochs. Initial results showed that the minimum intra-model bit-error was still rather high, greater than .1, so we increase training to ten epochs rather than five. We increased the number of epochs as we increased the scale of the noise added to allow for extra required training time for the model to converge to a new solution.

Although the noise scaling and length of training were changed significantly, we observed a similar pattern of results: the model worked well for several ratchet iterations but bit-error steadily increased with the number of times the model was ratcheted. After discovering the pivotal role of re-initializing the optimizer, in addition to then re-initializing the optimizer, we repeated the above experiment and re-initialized the final layers of the encoder and decoder rather than add random noise. Results improved drastically, as shown in Figure 4.15, and appeared similar to the SGM results: minimal collisions and low intra-model decoding bit-error rates. Specifically, convergence was much quicker, bit-error was very low for corresponding encoder and decoder pairs and high for non-corresponding pairs.



(a) Bit-error threshold (b) Diverging colormap

Figure 4.14. Bit-error between encoders and decoders from ratcheted models using random noise and not re-initializing optimizer.



(a) Bit-error threshold (b) Diverging colormap

Figure 4.15. Bit-error between encoders and decoders from ratcheted models using reinitialized encoder/decoder final layers and reinitialized optimizer.

4.4 Vulnerability to Steganalysis

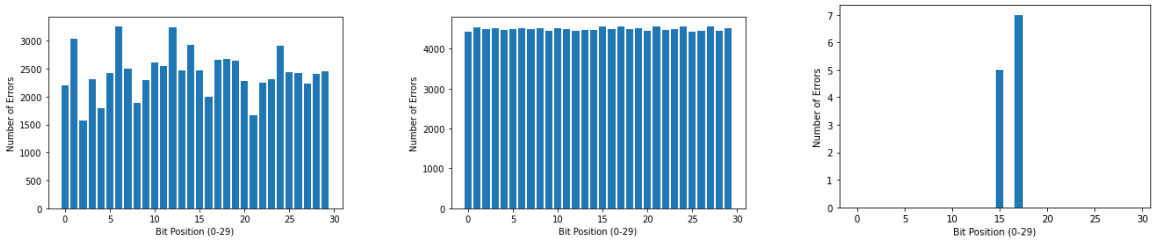
In this section we describe several experiments to evaluate our model's resistance to steganalysis.

4.4.1 *Determine Bit-Error Distribution*

In this experiment we sought to determine the distribution of bit-errors for a given test image over multiple models. For example, if we have a model that has a .1 decoding bit-error, for which bits are those errors happening? Is the third bit more likely to have an error than the second? In other words, we wanted to determine if the errors are independently distributed or if they follow some sort of pattern. Answering this question is important because it can have a significant effect on an adversary's ability to perform steganalysis and can tell us if interleaving in the form of adding error-correcting codes is necessary. If we determine that stego-images generated by the HiDDeN algorithm will cause non-uniform distribution of bit errors when attempting to decode the embedded message, then we will have created a simple steganalysis tool. If the the bit-error distribution is uniform, then the image in question was not likely created with HiDDeN.

For this test, pictured on Figure 4.16(a), we modified the previous encoding/decoding compatibility tests. We looked at one model over 3 epochs and tested all encoder and decoder combinations. For each encoder/decoder pair, we generated a random thirty-bit string, encoded our test image, decoded it and recorded the position of any decoding errors in a 30 length counter vector. This was performed for one thousand iterations on each encoder/decoder pair. We then took the counter vector and graphed it as a histogram, with the x-axis as a 0-29 bit position and the y-axis indicating the number of total errors experienced. Due to the way in which the experiment was conducted, the inter-model decoding bit-error , intra-model bit-error, and the observed decoding bit-error when decoding non-stego images is spread out across the histogram. It is apparent from the graph that the bit errors were not distributed randomly. Figure 4.16(b) contains results of an experiment decoding only non-stego images and testing the decoded string against randomly generated bit-strings. As one would expect, due to the fact that no actual encoding has taken place, the errors are distributed quite evenly across all bit positions reflecting the randomness of each new thirty bit string tested. Finally, the graph pictured in Figure 4.16(c) shows the distribution of bit errors in a single, "fully" trained model. In this context "fully" trained refers to a model that has an average bit-error of less than .05 when decoding stego-images that it has generated. Even though the bit-error rate is low, the errors that do occur are highly concentrated in certain bit positions, in this case, bits 15 and 17. The fact that both inter-model and intra-model bit errors appear not to be randomly distributed indicates that this model is likely

very vulnerable to steganalysis techniques that analyze the distribution of bit-errors.



(a) Inter-model and intra-model bit-error

(b) Bit-error for non-stego images

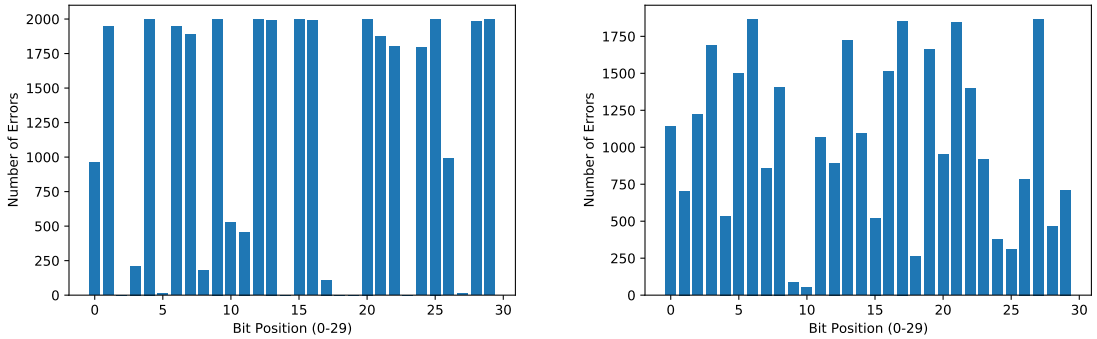
(c) Intra-model bit-error for single model

Figure 4.16. Comparison of relative bit-error position across multiple models (a), against cover images only (b), and within a single epoch/model (c).

After initial testing showed that bit errors were generally not distributed randomly, we performed several more experiments inspired by an adversarial approach. In this case, as described in Section 3.1, we assume that our adversary has passive access to all images sent via this communication channel and also to a functional steganographic, HiDDeN-based model, to include SGM and ratcheted models. Although our adversary may have access to various models, we assume that the adversary does not have access to the test model. In other words, the adversary does not know the weights/keys used for the tested pair of users' steganographic model. In this experiment scenario, the adversary can use its model to attempt to decode any images passed over the network. For example, it can take a known plaintext string as a stand-in for some secret message and test stego-images produced by other models, and by analyzing the distribution of bit-errors determine if the tested/decoded image is a stego-image or a cover image. Although the adversary could guess the specific message, such as a banned keyword, and test the decoded messages from random images against it, the success of the attack is not dependent on the choice of known plaintext. A successful attack does not give the adversary access to the original message, but instead the distribution of decoding bit-errors in a given image will tell the adversary if the tested image in stego-image or a non stego-image.

The following experiment is split into two parts. For the first part, we took a plaintext 30-bit string and encoded it using steganographic model A into 2000 random images. We assume that this string is known to the adversary. We then took a separate steganographic model,

model B, and used it to decode the stego-images produced by model A, and then calculated the decoding bit-error. We calculated the bit-error by testing the decoded value against the original plaintext string and recorded the bit position of each error. This tests whether an adversary who has obtained a legitimate copy of the overall steganographic model can viably use it to test whether or not steganography using the underlying algorithm has been used on a given image. We use a separate model for encoding, model A, and decoding, model B, in this experiment because we assume that the adversary does not have access to every unique solution/model weights that the algorithm can generate. The cover images are tested to see if there is a significant difference in the distribution of bit-errors in stego- vs. non-stego/cover images.



(a) Stego-images produced by separate model (b) Cover images only

Figure 4.17. Position of bit-errors in decoding known plaintext string in stego-images produced by another model compared versus cover images.

As can be seen from both graphs in Figure 4.17, bit-errors were distributed non-randomly in decoding both stego-images produced by another model and in decoding cover images, with no encoding whatsoever. Even though both sub-experiments showed non-random distribution, the bit-error distribution seen when decoding stego-images is significantly different from that seen when decoding cover images. These results are significant because they imply that an adversary could use this simple statistical analysis technique to determine whether or not an image is a stego-image. This property of the HiDDeN model, and potentially other ML-generated steganographic models is potentially a serious flaw as it essentially breaks the underlying model by giving an adversary a low-cost technique to

identify stego-images. Although the contents cannot necessarily be decoded, especially if the message is also encrypted, the main advantage of using steganography in the first place, covertness, is gone.

4.4.2 *Test Stego-Images Against Common Steganalysis Techniques*

In order to verify that our modified version of the HiDDeN model was still resistant to common steganalysis techniques, as the original developers of the algorithm claimed, we performed several steganalytic attacks against the model. For each test, we generated 20 total stego-images each encoded with a with random 30-bit string, specifically 10 stego-images were generated by each model, and 20 cover images and recorded the number of positive detections of steganography for each statistical technique. Testing was performed using steganography researcher Daniel Lerch's steganalysis program, Aletheia [42]. The title of the tables indicate how many separate models were used to generate stego-images. For example, the table titled "Two HiDDeN Models" shows results of statistical steganalysis techniques against 20 total stego-images, with 10 stego-images being generated by each model. The table titled "One Ratcheted Model" shows results for steganalysis against 10 stego-images from one model and 10 cover images.

Two HiDDeN Models		
Positive Detections		
Steganalysis Technique	Stego-images	Cover images
SPA	4	7
WS	4	6
Triples	2	4
RS	6	8

Two SGM-generated Models		
Positive Detections		
Steganalysis Technique	Stego-images	Cover images
SPA	1	1
WS	0	0
Triples	0	0
RS	2	3

Two Ratcheted Models		
Positive Detections		
Steganalysis Technique	Stego-images	Cover images
SPA	14	7
WS	12	6
Triples	16	3
RS	16	8

Although statistical techniques appear to be ineffective against the SGM models, initial results suggest ratcheted models appear to be vulnerable. In our experiment testing statistical attacks against stego-images generated by two ratcheted models, these attacks were nearly twice as likely to detect the use of steganography in stego-images than in cover images. Because of these initial results we ran another test, this time testing stego-images produced by one ratcheted model. Results show similar detection rates between stego- and cover images, suggesting the results from the previous experiment may be an aberration. However, in order to truly verify our model's robustness against steganalysis, further testing would be required.

One Ratcheted Model		
Positive Detections		
Steganalysis Technique	Stego-images	Cover images
SPA	7	8
WS	5	6
Triples	3	4
RS	10	8

CHAPTER 5:
Conclusions, Challenges and Future Research

Although our results showed potential for the successful implementation of an adaptive ML generated steganographic model, it also identified several key challenges. Our research has shown ways in which some of these challenges can be mitigated but continued research is required. The following section explores observations and conclusions from our research.

5.1 Conclusions

Our research results highlight several main areas of concern to include model message length, the importance of implementing error-correcting codes, and our model's weakness to basic statistical steganalysis techniques.

5.1.1 Message Length

One major shortcoming of our chosen model, the HiDDeN model, developed by researchers at Stanford University [25], was identified very early on in the research process; the inability to train a usable model on longer message lengths, greater than 128 bits. Our stated requirement of 280 characters (2240 bits) far exceeds the message lengths we were able to successfully train even after weeks of training on modern GPUs. Without an ability to train a model than can reliably encode and decode more than a word or two, the model would not meet our requirements. However, this approach may be suitable for some alternate channel with high message frequency and small payloads, such as preview thumbnails.

Therefore, we chose to investigate whether or not it would be effective to slice an image into smaller tiles and use a smaller message length model to encode data in each tile. The recipient would then slice the sent stego-image into the same tiles and decode each one in a certain order, and concatenate the data to decode the original data. Our results showed that depending upon the entropy of the given image, this approach produces some visual artifacts, although they are sometimes quite hard to see. This is due to the specific technique the model has learned to use to encode data. Parts of an image which contained mostly a single color (low entropy) were quite susceptible to visual artifacts because the majority

of the modified bits were in the middle of the image, producing something similar to a cross-hatch pattern over multiple tiles, which likely makes detection by steganalysis easier. Spectral analysis of the image may reveal a signal at the tiling frequency. This effect could be further mitigated by restricting the images that could be used with this model to those with high entropy. However, that would negatively impact the usability of this model. Another potential solution is to add noise around the edges of each tile. Because most of the encoded data is located in the middle of those low entropy tiles, this approach should not have a great effect on the decoder's ability to accurately decode the original bit string. However, further testing is required to determine if commonly used edge softening techniques affect decoding bit error in a slicing and tiling solution.

5.1.2 Implementing Error Correcting Codes

The use of error correcting codes can be useful on two fronts: further reducing intra-model bit error, and creating a more uniform distribution of bit errors. By measuring the relative frequency of bit errors experienced when attempting to decode a stego-image generated by a HiDDeN model, we showed that the the error distribution is far from uniform. On the other hand, when decoding a non-stego image against a known plaintext, the model's bit error distribution was quite different and not uniform. Assuming this software is open and available to everyone, a threat actor could easily obtain their own unique model. Once an attacker obtains their own model, they can then conduct the attack described in Section 4.4.1 to determine whether a given image is a stego-image produced by the HiDDeN algorithm. This technique does not require the attacker knows the original bit string used for encoding. The attacker only needs to use a constant known plaintext in order to generate a bit-error distribution which acts as a kind of fingerprint for the use of steganography.

This steganalysis via observation of bit error distribution would be complicated by the use of an error correcting code. Error correcting codes use interleaving or permutation which scramble the bits over code words and thus affect the distribution of errors [43]. By first adding an error correcting code to a message and then encoding it with our model, we would then decode the message with our model and apply the error correcting code to reduce errors. Although this may reduce and even completely prevent most errors, assuming the distributed model already has a very low bit error, this still does not prevent an adversary with full remote access to a device from testing for the use of steganography. Therefore,

we must implement cryptography. Because steganography and cryptography have different goals, there is an opportunity to achieve the "best of both worlds" by combining them into one overall framework. Steganography provides covertness and cryptography provides confidentiality, integrity, and/or authentication, depending on the implementation.

5.1.3 Resisting Steganalysis

Our aim is to ensure that our framework is resistant to today's most widely used steganalysis techniques. Our initial testing showed that HiDDeN models and our SGM and ratchet-generated models appear resistant to statistical steganalysis. However, a major weakness was identified in the experiments exploring the model's bit-error distribution. Our results show that bit-errors from decoding stego-images, even from other models, are distributed in a much different way than the errors observed when decoding cover/non-stego images. The known plaintext statistical analysis experiment outlines a simple approach for identifying stego-images produced by the HiDDeN model. Without even exploring the multitude of other steganalysis techniques that exist, we have already identified a major flaw. In order to create a viable steganographic model, this weakness would have to be overcome. It is unclear whether or not this can be fixed without substantive structural changes to the model.

5.2 Challenges and Future Work

This section discusses key challenges and considerations for future work related to designing, building, and distributing a ML steganographic model for anti-censorship communication.

5.2.1 Maximizing Entropy: Avoid Collisions and Increase Security

Through our experiments, we learned that adding random noise to the final layers of our encoder and decoder was not as effective as completely re-initializing those layers in terms of convergence speed and avoiding collisions. However, due to the Pytorch implementation of this re-initialization, namely the "reset_parameters()" method that is called for the respective encoder and decoder final neural net layers, the specific re-initialization method is not obvious. According to Pytorch's Github code repository, calling the "reset_parameters()" method for a ConvXd or Linear neural network uses a Kaiming initialization, also known as He initialization [44]. He initialization uses a scaled uniform distribution that takes

into account non-linear rectifiers [45]. Python's documentation states that the random() function's underlying C implementation utilizes a Mersenne Twister, a commonly used Pseudorandom Number Generator (PRNG) [46]. The documentation also states that the Mersenne Twister is both "completely deterministic" and "completely unsuitable for cryptographic purposes" [46]. Although the implementation of this PRNG in our steganographic framework is not cryptographic in nature, we seek to maximize security to the greatest extent possible and thus want to avoid using Python's random() function. Python documentation suggests using the included "os.urandom()" function which it claims is suitable for cryptographic purposes [47]. The documentation claims this method utilizes at least one "OS-specific randomness source" to add noise to the random number generation process. Our framework would benefit from this enhanced security. Beyond Python's own built-in methods, a popular technology that uses natural randomness inherent in the integrated circuit manufacturing process are Physical Unclonable Functions (PUF)s [48]. PUFs utilize this variability to generate "device-unique, unclonable keys" [48]. Whereas Python's built-in "os.urandom()" function relies on software-based system entropy, PUFs rely on hardware.

By generating random numbers using additional sources of noise, such as from operating systems or hardware, we are patching a potential threat vector. This ensures the uniqueness of individually trained models' weights. As we treat the weights of our models as keys, we can think of the process to generate weights from random seeds, which in a ML context would be in the form of parameter initialization, as a sort of Key Distribution Function (KDF). By utilizing both a random seed and the output of PUF or software-based source of randomness as input to our KDF or parameter/weight initialization process, we can transmit those keys/weights directly, or if we wish to add a level of abstraction, transmit the underlying random seed and PUF output. Although less efficient than transmitting the weights directly, these pre-inputs can be used to initialize and train a congruent model on another user's device, thereby creating a symmetric communications channel. In this context, by symmetric communications channel, we mean two users that have identical steganographic models, allowing them to encode, send, and decode stego-images between themselves with an extremely low chance of an adversary being able to either detect the use of steganography or read the contents of the encoded message. In order to protect the contents of our message, we must incorporate cryptography.

5.2.2 Implementing Cryptography

In the event that the adversary obtains our unique model, including its weights, and the ability to ratchet the model identically to the users, our steganographic solution has failed. The entire point of using steganography is to send secret messages without being detected. In this event, although the model may be compromised, there is still hope to prevent our adversary from reading the contents of our messages by implementing cryptography. In general, cryptography has been studied more extensively than steganography and certain algorithms and protocols possess provable security guarantees.

5.2.3 Developing an Integrated Framework: Steganography, Error-Correcting Codes, and Cryptography

In the section discussing error correcting codes above, we began discussing a protocol for two users, Alice and Bob, to communicate with our model. We discussed adding error correcting codes prior to embedding the secret message in the cover image. In the cryptography section, we discussed basic concepts but did not directly discuss how in might be implemented with steganography. Below, we will outline the steps necessary to integrate cryptography within our model.

This model makes two main assumptions: the first is that Alice and Bob have already negotiated a value to use for an initialization seed with which to ratchet their model to the same solution. In other words, they need some type of key agreement protocol. Once they agree upon an initialization seed, they can then train for a set number of epochs, allowing the model to re-converge to a solution. Only at this point will Alice and Bob have two nearly identical, fully compatible stego-models. The second major assumption our above model makes is that implementing cryptography will not require any extra round trips of data exchange. This assumption is false. In order to accomplish the cryptographic key agreement process, several messages need to be exchanged before the content can be encrypted. The following is a list of the necessary steps to incorporate both ECC and cryptography with our model:

1. Alice, the sender, drafts the message to be embedded
2. Alice encrypts the message.
3. Alice adds error-correcting code to the message.

4. Alice encodes encrypted message with error code into cover image with steganographic model shared between Alice and Bob.
5. Bob decodes the message using steganographic model he shares with Alice.
6. Bob applies ECC to message to correct errors.
7. Bob decrypts the original message.

5.2.4 End-Device Computing Power Limitations

A major challenge to the implementation of our framework is the computational power involved in training. In our experiments, we utilized modern, high-end GPU's. Although initial experimentation proved successful, due to time and prioritization of potential experiments, mobile devices were not tested. However, Pytorch does offer a mobile platform for iOS, Android and Linux [49]. In order to verify that our model is feasible, future experiments should involve training and testing our model on mobile hardware. Throughout our experimentation, it became obvious how important it is to optimize training techniques to our hardware. For example, selecting an appropriately large batch size can significantly reduce training time. For some of our early experiments, we relied in the default programmed batch size that was significantly smaller than what our GPU memory could hold. Therefore, the bottleneck we experienced was likely input/output related. We were only using a fraction of the available GPU memory. Once this bottleneck was identified, subsequent training was significantly faster. All this goes to show the importance of appropriately matching and optimizing software techniques for the hardware being used. By appropriately optimizing our techniques, we significantly reduce training time. In the case of increasing the batch size, changing one parameter led to a massive increase in performance. The same attention should be paid to experiments on mobile devices. Without dedicated GPUs, Compute Unified Device Architecture (CUDA) will no longer be a viable option and the original code may have to be modified. It may even be the case that the computation required by our model framework is too high to be run on a single device and may have to be run in parallel on multiple devices.

5.2.5 Key Space

In our experiment testing the feasibility of the SGM, we showed that for 100 models generated, there was a minimum of 8 collisions in the experiment set. For the random noise

addition approach, far more collisions were observed. Although the specific bit-error values that qualified as collisions were rather close to the threshold, values approaching .4 but still less than .4, 8 collisions out of 100 total is still unacceptable. For example, if we were to implement SGM at a large scale, increasing the total number of communicating pairs of users to 100,000, to provide pairs of users unique keys/weights for their models, a minimum of 8,000 collisions would be produced and likely far more, as the available "key space" (number of possible unique weights generating distinct models) is constant.

In order to evaluate our model's ability to scale, one might take a systemic approach by first exploring the theoretical bounds of the "key space" and justify that with testing. Depending on the initial results obtained from this approach, heuristics-based testing to estimate the key space may be useful, beyond the type of limited-scope model generation feasibility experiments described in our research. Estimating the key space for this model would require creating many individual models from scratch, training them until convergence, and then evaluating the range of different values obtained for the weights of the final layers of the encoders and decoders. This is a way of determining what kind of values result in a "usable" model (i.e. a model with intra-model bit-error < .05). It may be the case that the key space is too small to accommodate large number of users, or pairs of users. If it is the case that the total possible number of "unique" models is small, say less than 1 billion, then just using the weights of the final layers of the encoder and decoder as keys is insufficient. It would be possible to also use weights from other layers of the encoder and decoder to increase the total key space; however, this would increase required training time to generate a new model. Future work exploring generating new models with the HiDDeN algorithm via re-initialzing the final layers of the encoder and decoder should first explore the overall key space within those final layers.

5.2.6 Key and Model Distribution

One of the main difficulties in both steganography and cryptography is key distribution, or the process of securely sending keys to users. Historically, in cryptography, this problem was solved using an alternate, secure communications channel to send cryptographic keys that would be used to secure another communications channel. For example, a courier with a message could physically deliver a message containing a secret key that could be used by two parties communicating via telegraph to create a secure communications channel. In 1976,

this out-of-band, cryptographic bootstrapping problem was initially solved by Whitfield Diffie and Martin Hellman when they developed an asymmetric protocol that required users to exchange values across a public, insecure channel and then combine them using modular exponentiation, enabling each user to generate the same shared secret key [50].

Given that our steganographic model uses weights as "keys," a similar approach may be possible to send the keys across a public channel. However, it is important to keep in mind that in this case, the key are the weights of our model, which directly determine exactly how messages are encoded into cover images. A suitable protocol would require that the final key/weights obtained by each user not only be equivalent, but also result in a valid ML solution. In this context, the weights are more than just a shared secret: they are the recipe for exactly how the algorithm is implemented and a key exchange protocol that somehow mixes inputs from two users must result in a usable set of weights that correlates with a valid steganographic solution that encodes and decodes images with a low bit-error.

Another possible alternative to an asymmetric steganographic key exchange protocol is to send the keys in a secure, out-of-band communications channel, such as an encrypted email, message or even a physical, hand-carried message. However, many authoritarian regimes do not allow the use of encrypted email or messaging and if one can deliver hand-carried messages in person, it begs the question of why they would be using digital steganography to pass secret messages to begin with.

5.2.7 Ensuring User Access to Download and Use Application

As discussed in the first chapter, the main challenge to the spread and use of privacy-enabling mobile applications in mainland China is that the Chinese government is likely to eventually ban such an application. Signal, WhatsApp, Facebook Messenger, and in February of 2021, Clubhouse, have all been banned by the Chinese government and are no longer available for download or use in Mainland China. In the case of our steganographic framework, we require running a plugin-type application alongside/on top of an existing messaging application. If we can first overcome the bootstrapping problem of enabling a user to download our application to his or her device, ideally via some easily accessible application marketplace such as the Google Play Store or the Apple App Store. Other potential solutions could be Peer to Peer (P2P) file-sharing or even in-person sharing of physical storage devices that

contain the program.

5.2.8 Building and Integrating the Mobile Application

Although we have discussed many technical challenges to building a mobile steganographic communications solution, we have not yet discussed the inherent difficulty of both building a usable mobile application and successfully integrating this application to an existing one. Specifically, for our model requirement to be fully satisfied, our mobile application would function as a plugin for an already widely used messaging application adding functionality while utilizing the communication backbone of that application.

One of the major issues we face with the development of such an application is that many messaging platforms do not support "unofficial" plugins. For example, published WeChat policy states that "emulators and unofficial plugins endanger security and are strictly prohibited" [51]. Therefore, at least in the case of developing a solution compatible with WeChat, we would either need to subvert the mechanism that detects the use of the unofficial plugin or separate our mobile application entirely, in terms of inter-process communication and Application Programming Interface (API). Both approaches introduce their own complications. The first approach, to circumvent the detection of unofficial plugins by the original application developer, may involve a significant software reverse engineering effort as the mechanisms to detect such behavior are likely unpublished and not obvious. The main downside is that significant time and effort are required just to understand how the detection process takes place. If this can be determined, then there is still the question of whether or not it can be circumvented.

The second approach avoids the unofficial plugin problem by building an application on top of WeChat that uses WeChat's user interface as an API rather than directly interacting with the AI itself. By simulating a user, the application can interact directly with WeChat without interfacing with the official API. The program could automatically enter text or attach selected images in WeChat. This second application would produce a stego-image to be sent via a messaging application, and then automatically attach and send in the messaging application. If we successfully implemented this approach, in order to send a stego-image the user would first need to open our application, select a cover image and enter the text to be encoded, press some kind of a button to produce and save the stego-image, and the

program would then automatically open the main messaging application, attach the stego-image, and send it to the desired recipient. The recipient would then need to open our mobile application. The mobile application would automatically copy the received stego-image from the messaging application to our mobile application, which would then decode and display the message. Because our application is completely separate from the main messaging application, it would require a programming solution that relies interacting with the target application's user interface, which is likely much less efficient and potentially more complex that interacting directly with the API.

Finally, we have not even begun to address the inherent challenges in developing a mobile application. In addition to considering the audience/user needs, we also need to consider OS compatibility, battery use, performance and secure software development. Although these challenges have all been mentioned or described at least indirectly, properly addressing them is not trivial.

5.2.9 Concluding Remarks

Our research demonstrated the potential for ML steganography to be used as an anti-censorship mechanism while also highlighting key challenges. Perhaps the greatest challenge to the implementation and widespread use of ML steganography is its apparent weakness to a statistical steganalysis technique that analyzes the distribution of bit-errors. Images can be checked in a limited amount of time with little computation, neutralizing the main covertness benefit of using steganography. Although some forms of ML steganography may be vulnerable to such attacks, more research is required to determine the extent to which other ML steganography algorithms are vulnerable.

Although an adaptive, ML steganographic communications framework is challenging to design and implement, it serves as an added security layer to more common encrypted messaging platforms. Encrypted messaging applications typically advertise their encryption capabilities and are blocked in many authoritarian countries. As citizens around the world become more tech-savvy and privacy conscious, embracing decentralized finance and communication networks, the demand for privacy-enabling anti-censorship technology will only increase. The development and proliferation of such technology would enable free and open communication in support of democratic values and would be beneficial for not

only the United States and like-minded nations, but also for anyone in the world who values individual freedom, no matter where they live.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] R. Sundara Raman, P. Shenoy, K. Kohls, and R. Ensafi, "Censored planet: An internet-wide, longitudinal censorship observatory," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[2] CitizenLab, "We chat, they watch: How international users unwittingly build up WeChat's Chinese censorship apparatus," May 2020. Available: https://citizenlab.ca/2020/05/we-chat-they-watch/

[3] Z. Lu, Y. Jiang, C. Lu, M. Naaman, and D. Wigdor, "The government's dividend: Complex perceptions of social media misinformation in China," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Honolulu HI USA: ACM, Apr. 2020, pp. 1–12. Available: https://dl.acm.org/doi/10.1145/3313831.3376612

[4] R. Staff, "Encrypted messaging app Signal stops working in China," *Reuters*. Available: https://www.reuters.com/article/us-china-tech-signal-idUSKBN2B8094

[5] B. Dowling and B. Hale, "There Can Be No Compromise: The Necessity of Ratcheted Authentication in Secure Messaging," Tech. Rep. 541, 2020. Available: https://eprint.iacr.org/2020/541

[6] C. Montag, B. Becker, and C. Gan, "The multipurpose application WeChat: A review on recent research," *Frontiers in Psychology*, vol. 9, 2018. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2018.02247/full

[7] Y. Chu and H. Ruthrof, "The social semiotic of homophone phrase substitution in Chinese netizen discourse," *Social Semiotics*, vol. 27, no. 5, pp. 640–655, Nov. 2017. Available: http://libproxy.nps.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=sih&AN=124637933&site=ehost-live&scope=site

[8] W. House, "U.S. strategic framework for the Indo-Pacific." Available: https://www.whitehouse.gov/wp-content/uploads/2021/01/IPS-Final-Declass.pdf

[9] K. Benner, "Parler says it sent the F.B.I. posts about threats to the Capitol ahead of Jan. 6." *The New York Times*, Mar. 2021. Available: https://www.nytimes.com/2021/03/25/us/parler-fbi-capitol-attack.html

[10] G. Clary, "Parler wasn't hacked, and scraping is not a crime." Available: https://www.lawfareblog.com/parler-wasnt-hacked-and-scraping-not-crime

[11] D. Shepardson, “U.S. panel asks FBI to review role of Parler in Jan. 6 Capitol attack,” *Reuters*, Jan. 2021. Available: https://www.reuters.com/article/us-usa-trump-parler-idUSKBN29Q2FS

[12] D. Townson, “Hundred Flowers Campaign (1956 - 1957) | A Dictionary of Contemporary History - 1945 to the present - Credo Reference.” Available: https://search.credoreference.com/content/entry/bkchist/hundred_flowers_campaign_1956_1957/0

[13] X. Li, *Civil liberties in China* (Understanding China today). Santa Barbara, CA: ABC-CLIO, 2010.

[14] W. House, *National Security Strategy*. Washington, DC: White House, 2017. Available: https://nssarchive.us/national-security-strategy-2017/

[15] ACLU, “Freedom of expression in the arts and entertainment.” Available: https://www.aclu.org/other/freedom-expression-arts-and-entertainment

[16] P. Gewirtz, “On ‘I Know It When I See It’,” *Faculty Scholarship Series*, Jan. 1996. Available: https://digitalcommons.law.yale.edu/fss_papers/1706

[17] D. Kahn, “The history of steganography,” in *Information Hiding* (Lecture Notes in Computer Science), R. Anderson, Ed. Berlin, Heidelberg: Springer, 1996, pp. 1–5.

[18] T. W. Ph.D., “S1E05 Prezbo cracks code.” Available: https://www.youtube.com/watch?v=DQBlq45c1T4

[19] F. A. P. Petitcolas, *Kerckhoffs’ Principle*. Boston, MA: Springer US, 2011, pp. 675–675. Available: https://doi.org/10.1007/978-1-4419-5906-5_487

[20] C. S. Division, “AES Development - Cryptographic Standards and Guidelines | CSRC | CSRC,” Dec. 2016. Available: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development

[21] D. P. Leech, S. Ferris, and J. T. Scott, “The economic impacts of the advanced encryption standard, 1996-2017,” National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST GCR 18-017, Aug. 2018. Available: https://nvlpubs.nist.gov/nistpubs/gcr/2018/NIST.GCR.18-017.pdf

[22] H. G. Schaathun, *Machine Learning in Image Steganalysis: Schaathun/Machine Learning in Image Steganalysis*. Chichester, UK: John Wiley & Sons, Ltd, Aug. 2012. Available: http://doi.wiley.com/10.1002/9781118437957

[23] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, “Digital image steganography: Survey and analysis of current methods,” *Signal Processing*, vol. 90, no. 3, pp. 727–752, Mar. 2010. Available: https://linkinghub.elsevier.com/retrieve/pii/S0165168409003648

[24] S. Chutani and A. Goyal, "A review of forensic approaches to digital image Steganalysis," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 18 169–18 204, July 2019. Available: https://doi.org/10.1007/s11042-019-7217-0

[25] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "HiDDeN: Hiding data with deep networks," *arXiv:1807.09937 [cs]*, July 2018, arXiv: 1807.09937. Available: http://arxiv.org/abs/1807.09937

[26] J. Hayes and G. Danezis, "Generating steganographic images via adversarial training," *arXiv:1703.00371 [cs, stat]*, July 2017, arXiv: 1703.00371. Available: http://arxiv.org/abs/1703.00371

[27] D. Lerch-Hostalot and D. Megías, "Unsupervised steganalysis based on artificial training sets," *Engineering Applications of Artificial Intelligence*, vol. 50, pp. 45–59, Apr. 2016. Available: https://linkinghub.elsevier.com/retrieve/pii/S0952197616000026

[28] "Clinton's Words on China: Trade Is the Smart Thing (Published 2000)," *The New York Times*, Mar. 2000. Available: https://www.nytimes.com/2000/03/09/world/clinton-s-words-on-china-trade-is-the-smart-thing.html

[29] C. Göbel, "Social unrest in China: A bird's-eye view," *Handbook of Protest and Resistance in China*, June 2019. Available: https://www.elgaronline.com/view/edcoll/9781786433770/9781786433770.00008.xml

[30] R. Zhong, P. Mozur, J. Kao, and A. Krolik, "No 'negative' news: How China censored the Coronavirus," *The New York Times*, Dec. 2020. Available: https://www.nytimes.com/2020/12/19/technology/china-coronavirus-censorship.html

[31] G. King, J. Pan, and M. E. Roberts, "How censorship in China allows government criticism but silences collective expression," *American Political Science Review*, vol. 107, no. 2, pp. 326–343, May 2013. Available: https://www.cambridge.org/core/product/identifier/S0003055413000014/type/journal_article

[32] J. Knockel, L. Ruan, M. Crete-Nishihata, and R. Deibert, "(Can't) picture this: An analysis of image filtering on WeChat moments," Aug. 2018. Available: https://tspace.library.utoronto.ca/handle/1807/94801

[33] J. C. Hernández, "30 years after Tiananmen, 'Tank Man' remains an icon and a mystery," *The New York Times*, June 2019. Available: https://www.nytimes.com/2019/06/03/world/asia/tiananmen-tank-man.html

[34] M. Roberts, *Censored: Distraction and Diversion Inside China's Great Firewall*, 2018, oCLC: 1085405020. Available: https://doi.org/10.23943/9781400890057

[35] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *arXiv:1801.03924 [cs]*, Apr. 2018, arXiv: 1801.03924. Available: http://arxiv.org/abs/1801.03924

[36] Signal, "Signal messenger: Speak freely." Available: https://signal.org

[37] L. Tian, D. F. Wong, L. S. Chao, P. Quaresma, F. Oliveira, and L. Yi, "UM-corpus: A large English-Chinese parallel corpus for statistical machine translation," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, N. C. C. Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, Eds. Reykjavik, Iceland: European Language Resources Association (ELRA), may 2014.

[38] ando-khachatryan, "ando-khachatryan/HiDDeN," Dec. 2020, original-date: 2018-12-10T13:39:23Z. Available: https://github.com/ando-khachatryan/HiDDeN

[39] C. Incorporated, "Graphics Interchange Format(sm) Version 89a," July 1990. Available: https://www.w3.org/Graphics/GIF/spec-gif89a.txt

[40] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[41] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. Available: http://arxiv.org/abs/1412.6980

[42] D. Lerch-Hostalot, *Aletheia*, Apr. 2021. Available: https://doi.org/10.5281/zenodo.4655945

[43] C. Berrou, "Introduction," in *Codes and Turbo Codes* (Collection IRIS), C. Berrou, Ed. Paris: Springer. Available: https://doi.org/10.1007/978-2-8178-0039-4_1

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," *arXiv:1912.01703 [cs, stat]*, Dec. 2019, arXiv: 1912.01703 version: 1. Available: http://arxiv.org/abs/1912.01703

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *arXiv:1502.01852 [cs]*, Feb. 2015, arXiv: 1502.01852. Available: http://arxiv.org/abs/1502.01852

[46] P. S. Foundation, “random — Generate pseudo-random numbers — Python 3.9.2 documentation.” Available: https://docs.python.org/3/library/random.html

[47] P. S. Foundation, “os — Miscellaneous operating system interfaces — Python 3.9.2 documentation.” Available: https://docs.python.org/3/library/os.html#os.urandom

[48] K. Horovitz and R. Kenny, “Intel FPGA secure device manager,” Intel Corporation, Programmable Solutions Group (formerly Altera) San Jose United States, Tech. Rep., Mar. 2018. Available: https://apps.dtic.mil/sti/citations/AD1052301

[49] Facebook, “PyTorch.” Available: https://www.pytorch.org

[50] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[51] WeChat, “What do I do if my WeChat account has been blocked?” Available: https://help.wechat.com/cgi-bin/micromsg-bin/oshelpcenter?opcode=2&lang=en&plat=ios&id=1706082IVB7r17060873E3u6&Channel=helpcenter

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California